La primera revista del mundo que enseña la programación de videojuegos

# CODIGO JUEGO

975 pts

Año 1, Número 2, Julio 1994

Para todo tipo de ordenadores personales

Artículo

## Power PC
El futuro que se acerca

Nueva sección

## El Buen Soft
Guia del Shareware y DP

Dominio PÚBLICO

Comparativa

## Microsoft Macroassembler
La potencia de trabajo

Shareware

APRENDE A HACER TU PROPIO JUEGO CON MOOKIE

Cursos

Lenguaje C
Ensamblador
MIDI

## Falcon 030
Un ATARI muy especial

00002
8 424094 009756

Ajatel S.L.

CODIGO JUEGO

Editorial: AJATEL S.L.

Director: Ramón López 340 25 66

Subdirectores: Joan Esquerra (96)
Miguel A. Lobato

Directora de Arte: Maite López FRAN

Colaboradores: Chema Ginés
Manuel Gallego
José A. Brihuega Parodi
Eduard Sánchez
Roberto Cruz
Alfonso S. Cachafeiro
María Jesús Cano Rueda
Juan Manuel Rial: ATARI

Asesoría Jurídica: Esther Lobato

Fotografía: Manuel Gallego Palacios

Corresponsales USA: Timothy Conard
Luis Vázquez
Vicente Santos

Jefe de publicidad: Joan Esquerra
ASINFO: Teléfono/FAX: 956-813129

Subscripciones: Miguel L. Moreno
Teléfono/FAX: 956-840246

Filmación: Texto y Color
Paseo de las Delicias, 15
Teléfono 956-332912
Jerez de la Frontera

Imprime: GRAFIBÉRICA
Marqués de Cádiz 25
Jerez de la Frontera, Cádiz
Teléfono: 956-185500

Disquetes: KAO
Teléfono: 93-7182287

Distribuye: DISPAÑA S.L. S.C.
Avenida Valdelaparra 27
Edificio II, Local 9
Alcobendas, Madrid
Teléfono: 91-6620602

ISSN: 1134-0975
Depósito Legal: CA-286/94

Código Juego
Apartado de correos 114
Rota (Cádiz) 11520
Teléfono/FAX: 956-840246

Código Juego no se hace necesariamente solidaria de las opiniones vertidas en los artículos firmados. Está prohibido cualquier tipo de reproducción por cualquiera que sea el medio, sin permiso expreso del editor.

# SUMARIO

## ESTE MES...

EDITORIAL 4
NOTICIAS 5
CARTA DEL LECTOR 14
EL TINTILLA 15
ASI SE HACE UN... 16
EL JUEGO DEL MES 22
EL LIBRO DEL MES 28
LA VGA (II) 30
ASSEMBLER 68000 40
EL MSDOS 58
LA TRIQUIÑUELA 62
EL BUEN SOFT 63
CONCURSO 42
EN EL PROXIMO 67

**Desde la redacción de la revista queremos daros las gracias por todas las llamadas, cientos de ellas, que hemos recibido a lo largo de este mes, dándonos aliento en la publicación de Código Juego. Sois muchos los que nos habéis hecho saber vuestro apoyo incondicional, alegando que una publicación como la nuestra hacía falta en el mercado informático. También, queremos haceros saber que vuestras críticas no caen en saco roto y que, por ello mismo, podéis comprobar que el número de este mes ha cambiado sustanciosamente, intentando llegar a vosotros con la máxima calidad.**
**Nuestra promesa de seguir mejorando sigue en la mente de todos los que componemos el personal de Código Juego, por lo que la revista irá a más cada mes.**
**Un abrazo a todos.**

54

PowerPC™ 601

# Power PC

## A toda potencia

Comentamos las características de la nueva maravilla de IBM, Motorola y Apple.

# Cursos: PC

## Ensamblador

## Lenguaje C

## MIDI

¿Quién dijo que segundas partes nunca fueron buenas?. Continuamos con el estudio de estos lenguajes.

43

ATARI

# Falcon 030

Los amantes de la buena música, los sonidos y los gráficos no deben olvidar leer este artículo sobre uno de los mejores ordenadores para este fin.

51

LOS SONIDOS DEL PC

NUEVO CURSO

ENTREVISTA

20

## IGOR: Objetivo UIKOKAHONIA

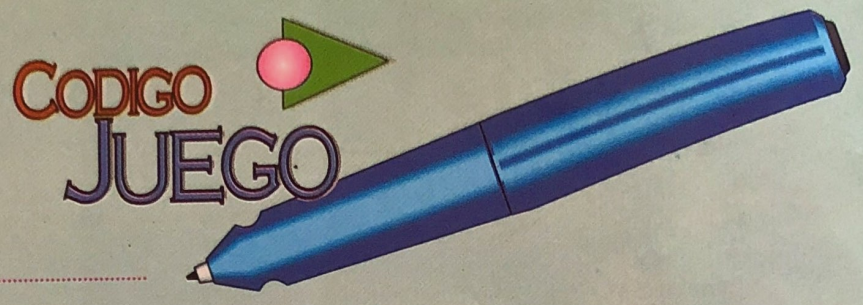
Primera aventura gráfica de origen español. Hablamos con Péndulo Studios. Sus creadores.

46

## MICROSOFT MACROASSEMBLER

*Analisis interno*

# Editorial




El otro día, mientras paseaba por un gran centro comercial, pasé cerca de la sección de informática. Tras aquellas grandes y limpias cristaleras vi que un muchacho, de unos 10 años de edad, tenía sujeto algo entre las manos al tiempo que se movía de una forma tan extraña que me llamó la atención.

No acertaba a adivinar que ocurría, así que, pasito a pasito, me acerqué como una serpiente a su presa. Cuando estuve lo suficientemente próximo a él, alcé la mirada y comprobé que, aquello que tanto estrujaba, era el mando de una consola de vídeojuegos y que lo que miraba, tan absorto, era la pantalla del televisor donde ésta estaba conectada.

Tan ensimismado se encontraba, que incluso tan cerca de él, no me advirtió, o si lo hizo no le importé lo más mínimo, por lo que aproveche la ocasión para estudiar sus reacciones detenidamente. Se movía cómo si lo hubiesen endemoniado. Se retorcía, sacaba la lengua, los ojos brillantes, blasfemaba de viva voz, daba zapatazos, y en ocasiones miraba hacía atrás buscando a su compañero que se divertía con otra consola.

Me hubiese quedado allí el resto del día, porque era digno de admirar como se divertían.

Supongo que, cuando un artista de escenario completa su actuación, y recibe el gran aplauso del público, debe sentirse de igual manera que yo ese día, plantado frente a aquel pequeñín encantador. Y es que, ver como un niño es capaz de disfrutar con lo que nosotros, los programadores de vídeojuegos, creamos es el mejor de todos los aplausos que podamos recibir. Y la verdad, no hace falta más.

Todo el trabajo y esfuerzo que cuesta desarrollar un vídeojuego, queda compensado, cuando ves que una sola persona disfruta con él. Aunque sólo sea un ser sobre toda la faz de la tierra. Y os puedo asegurar que aquel muchachito disfrutaba como un cosaco.

Por ello, cuando escucho a todos esos detractores de los juegos de ordenador, que argumentan que producen malos y desafortunados efectos secundarios en la juventud, me entristezco. Y es que, desde mi punto de vista, el juego los entretiene -alejándolos de las calles que cada día están peor-, les activa los reflejos, los hace competitivos... Incluso les enseña a ahorrar, haciendo que se priven de otros vicios para conseguir el dinero suficiente con el que pagar ese último juego maravilloso, que salió para la Super Nintendo o la Megadrive.

Y es que, desde que murió Walt Disney, los que programamos vídeojuegos hemos tomado el relevo, salvando

## SUMERGIMOS AL JUGADOR EN OTRO MUNDO

las diferencias, de lo que es la creación de ilusión. Nosotros, al igual que el genial creador de dibujos animados, tenemos la posibilidad de hacer que la persona se sumerja, aunque sólo sea durante un rato, en un mundo diferente lleno de acción, habilidad, inteligencia, destreza, terror, etc.

Por ello, animo a todos los que leéis la revista, a que estudiéis un poquito y os lancéis a la creación de vuestro propio vídeojuego en cuanto podáis. La mejor forma de aprender es equivocándose. No temáis, ya mejoraréis.

Por otro lado, espero que la revista os esté gustando.

Como podéis comprobar, ya en este número, contamos con una sección ENSAMBLADOR 68000 (Atari, Amiga) un poco más amplia y eso no es nada, por que va a seguir creciendo.

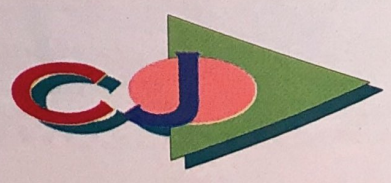
Los 3 cursos estrellas (Ensamblador, C y MIDI de la parte PC) os traen más cosas y nuevas lecciones. Comentamos también uno de los macroensambladores más completos del mercado; "Microsoft Macroassembler 6.0", a la par que seguimos la explicación sobre el depurador estándar del MS-DOS y la tarjeta VGA.

Además, nuestro juego particular MOOKIE va tomando cuerpo. Los listados incluidos en la revista anterior se explican ahora a fondo para que los entendáis perfectamente.

Inauguramos una interesante sección que nos va a acompañar durante los meses venideros; "El buen Soft". En ella comentaremos los programas sharewaré y de dominio público que están disponibles cerca de todos nosotros y que ni siquiera sabemos que existen. Estos, siempre estarán relacionados con la programación de C y Ensamblador, así como todos aquellos que puedan servir de ayuda para la misma. Nuestro experto en estos temas se hace cargo de ello.

También, desde aquí, desde las páginas de Código Juego, queremos dar las gracias a la cadena de radio ONDA CERO, por la ayuda que nos han prestado para la promoción de la revista. Desde el primer momento, el programa Bienvenido Mr. Chip y la emisora local Onda Cero Rota, se interesaron por nuestro proyecto y decidieron, desinteresadamente, ayudarnos y empujarnos un poco hacía arriba en esta difícil tarea de lanzar una nueva revista informática. Gracias de corazón.

Ramón López
Director de Código Juego



# FLASH Noticias

❑❑❑

Novedades Atari.

La compañía Compo Software ha presentado un compatible Atari TT llamado Medusa T40. El Medusa está basado en el 68040 de Motorola, con una velocidad interna de trabajo de 64 Mhz y con una frecuencia de reloj externa de 32 Mhz. La configuración básica vendrá con 8 Mb de memoria RAM, una tarjeta de vídeo ET4000 con 1 Mb de memoria, un disco duro IDE de 300 Mb, TOS 3.06, una tarjeta E/S ST, teclado tipo PC, ratón y disquetera HD. La tarjeta ST E/S contiene un puerto DMA, dos RS232, un centronics, MIDI, interface IDE y un slot ISA. Opcionalmente dispone de un interface SCSI 1 y una tarjeta gráfica de 32 bits color real con 4 Mb de memoria y una resolución de 1100x900 en 24 bits.

Especificaciones : 68040 (64/32 Mhz) FPU y PMMU interno 26 MIPS 4.5 MFLOPS Bus de datos 32 bit Bus de direcciones 32 bit 6 slot RAM ampliable a 128 Mb en placa base y hasta 4 Gb con tarjeta. Capacidad ROM: 0.5-2 Mb

Otra novedad de esta empresa es la tarjeta Falcon Afterburner, para el ordenador Atari Falcon030. Este acelerador posee el 68LC040 (33/66 Mhz) y una memoria FastRAM hasta 64 Mb. Esta provisto un bus PCI para tarjetas gráficas.

❑❑❑

Como era de suponer ya están apareciendo los programas para Atari Falcon que aprovechan el DSP no sólo para música.

Entre ellos se puede destacar "VoiceMail". El programa convierte el Falcon en una contestador automático inteligente. Este software junto con un poco de hardware permite dejar grabado mensajes privados para distintas personas o departamentos. Estos mensajes podrán ser oidos después por la persona en cuestión con o sin clave de acceso. (Precio aproximado : 35.000 ptas).

❑❑❑

Otro programa que aprovecha el DSP es la nueva versión de Studio Photo : Studio Photo DSP. Esta nueva versión aprovecha el DSP para tratamiento de imágenes y funciones de filtro. (Precio aproximado : 20.000 ptas, Studio Photo Professional : 35.000 Ptas)

❑❑❑❑❑❑❑❑❑❑❑❑❑❑❑❑❑❑❑❑


BitVision

San Jose Artesano 5
03500 Benidorm
Tel/Fax : 96-586 34 15

**Falcon Tower System**

**4 Mb RAM, DD 170 Mb, Teclado castellano PC 102 teclas, F.A. 230 W, 195.000 Pts**

**IVA incluido**

**Acelerador 32/40 Mhz opcional**

**Configuraciones segun su necesidad**

| | |
|---|---|
| Falcon 030 4/64 | 149.000.- |
| Calamus S 2.0 | 29.900.- |
| VideoMaster F030+RGB | 34.000.- |
| Scooter PCB | 29.900.- |
| Scooter PCB color | Consultar |
| Tarjeta Ampl. F030 | 12.000.- |
| Acelerador 32 Mhz F030 | 15.000.- |
| Acelerador 40 Mhz F030 | 22.500.- |
| Morpher 1.1 | 15.900.- |
| Pure C 1.1 (Librerias Falcon) | 38.000.- |
| Atari Works | Consultar |
| Tableta grafica Tabby A5 | 15.900.- |
| DA´s Vektor | 25.000.- |
| DA´s Vektor Pro | 40.000.- |
| DA´s Picture | 29.900.- |
| Programador EPROM | 14.500.- |

IVA no incluido.

# Subscribirse

# a

# CÓDIGO JUEGO

## tiene ventajas

**10 %** de descuento del precio de portada

**5 %** de descuento de los gastos de envío

**1** camiseta estupenda de regalo

Pedidos:
Teléfono: 956-813129
FAX: 956-840246
Correos: CODIGO JUEGO
AJATEL S.L.
Apdo 114
11520 Rota (Cádiz)

Ajatel S.L.

Parte 2

# Curso de Lenguaje Ensamblador Para PC

**Uno de los temas que mejor debe dominar todo buen programador de lenguaje ensamblador es aquel que concierne al microprocesador. Sin estos conocimientos, nuestros programas jamás podrán ser todo lo compactos y potentes que nosotros queremos. Y lo que es más importante, nunca tendremos una idea clara de por qué las instrucciones hacen lo que hacen.**

## El microprocesador

Como ya apuntamos en el número anterior de CJ, el microprocesador es la pieza clave del hardware de nuestro ordenador. En él, se agrupan las funciones principales de control y ejecución. Sin su participación, el PC sería como un rebaño de ovejas sin pastor, cada componente por su lado, en definitiva, un completo caos.

## Un carpintero muy especial

Para explicar como funciona el microprocesador, vamos a echar mano a una pequeña historia que le ocurrió a dos buenos amigos.

Supongamos que queremos que nos hagan una repisa de madera para poner nuestros libros de informática, porque estamos hartos de que estén siempre tirados por la habitación.

Nuestro amigo, Micro, es un tío muy apañado, famoso por hacer todo tipo de cosas muy rápido y muy bien, o sea, un verdadero manitas.

Le pedimos que, por favor, nos haga la repisa. El nos contesta que, aunque es un tío muy hacendoso, no sabe gran cosa de carpintería, así que nos pide que le hagamos un plano con todas las instrucciones detalladas, paso a paso, de como hacer el mueble en cuestión. Nosotros accedemos gustosos, ya que sabemos que nunca lo podremos hacer tan rápido y tan bien como nuestro amigo Micro.

Realizamos, cuidadosamente, el plano con todas los pasos a seguir. Las instrucciones son tan claras, que incluso un niño podría seguirlas.

Micro, coloca el plano sobre su gran mesa de trabajo y pone a mano todas las herramientas que sabe que le van a hacer falta. Ya está todo listo.

## Manos a la obra

Nuestro carpintero improvisado va leyendo cada instrucción, piensa como debe hacerla o ejecutarla y finalmente, tras meditar como la debe realizar, la lleva cabo finalmente. Terminada esta parte, lee otra instrucción, la interpreta y la ejecuta, y así sucesivamente hasta que el mueble está terminado.

## La realidad

Bien, analicemos este símil para ver sus analogías con la manera en que trabaja el microprocesador.

Nuestro amigo el carpintero, es el que, como habéis podido comprobar, realiza todo el trabajo, pero siempre se guia de nuestro plano de instrucciones. Sin éste, se podría perder y acabar haciendo una mesa o un sofá.

Lo mismo ocurre con la CPU, sin el programa, que se correspondería con el plano de nuestra historia, y a pesar de ser muy habilidosa, no sabe que debe hacer y en que orden debe ir haciéndolo. Micro, nuestro atareado amigo, sería en realidad el microprocesador. El lugar donde coloca el

plano/programa, esa gran mesa de trabajo, es la memoria de ordenador y , finalmente, las herramientas de las que hecha mano para llevar a cabo su tarea serían los distintos chips de apoyo que se encuentran en el conjunto del hardware del ordenador.

La CPU, como también se conoce al microprocesador, busca en la memoria la instrucción. A continuación, la decodifica para entenderla. Seguidamente, la ejecuta y, en último lugar, guarda los resultados de la operación.

Estos son los pasos básicos que el procesador realiza, para que todos los entendáis:

1) FETCHING: Ir a buscar y traer. La CPU va a la memoria y busca la siguiente (o la primera) instrucción a ejecutar.

2) DECODING: Decodificar. La instrucción es traducida, por el microcódigo del procesador, en la secuencia de pasos que deberá seguir éste para la correcta ejecución de la instrucción.

3) EXECUTING: Ejecutar. La instrucción, una vez decodificada, se ejecuta finalmente.

4) WRITING: El resultado de la ejecución se guarda en el lugar correcto. Por ejemplo, si sumamos un número con otro, el resultado de esta operación debe ser guardado para su uso posterior. Sino es así se perdería y la ejecución de nuestra instrucción no tendría sentido.

Una vez que el último paso se completa, la CPU, siempre incansable, vuelve a realizar todo el proceso con la siguiente orden de nuestro programa.

## El puntero de programa

Para que nuestro amigo Micro no olvide cual es la siguiente instrucción a ejecutar, siempre tiene la precaución de anotar cual es. Para ello utiliza una pizarra donde apunta el número de orden de ésta y, una vez que la termina, la borra y anota la siguiente a ejecutar y así hasta el final. El microprocesador hace lo mismo, llamando a su particular pizarra con el nombre de IP o Instruction Pointer (Puntero de instrucción). Siempre tendrá anotado en esta pizarra en número de la siguiente instrucción a ejecutar.

## Las pizarras

Una cosa que se nos olvido comentar de nuestro habilidoso compañero es que, para ayudarse en su trabajo, usa varias pizarras. A parte de la pizarra IP, descrita anteriormente, la CPU usa otras, donde va anotando los resultados de las operaciones, siendo éstas de vital importancia.

Por ejemplo, si queremos sumar un número con otro, como en nuestro ejemplo anterior, la CPU traerá el primer número de la memoria y lo anotará en una pizarra. A continuación trae el siguiente dígito y lo suma al primero, para acabar escribiendo el resultado en la pizarra de nuevo.

Veremos todas estas más adelante y aprenderemos el cometido especial que el microprocesador da a cada una de ellas.

**El microprocesador engloba todas las funciones importantes del ordenador.**

## Las instrucciones

Cuando le hicimos el plano a Micro, las instrucciones para completar la construcción de la repisa eran, poco más o menos así:

1) Cortar madera de 1 cm de grosor con medidas 40x 80 cms
2) Lijar madera
3) Apuntillar madera con listón
4) Pintar listón
5) etc...

Sin embargo, como vimos en el anterior artículo de este curso, la CPU real no sabe español, desgraciadamente. Pero, para ser francos, tampoco entiende inglés ni japonés. Unicamente, es capaz de entender su propio lenguaje; los unos y los ceros (1 y 0). Por ello, las ordenes Cortar, Lijar, Apuntillar y Pintar le son desconocidas.

Lo mismo ocurre con el programa que tuviese ordenes como SUMAR, RESTAR, SENO, COSENO, DIVIDIR, etc. Por ello, usamos los programas ensambladores y compiladores a los que hacíamos mención en CJ 1. Las ordenes o instrucciones, una vez traducidas con antelación sí pueden ser decodificadas, cosa que no nos cansaremos de repetir.

## El banco de trabajo

Una vez que hemos explicado, por encima, la forma en que el microprocesador trabaja, es primordial que conozcamos cual es el verdadero formato de su banco de trabajo; la memoria.

Como sabemos ya, en la memoria, que era la mesa de nuestro amigo el carpintero, es donde se sitúan los programas (secuencias de instrucciones que guían a la CPU en la manera que debe trabajar).

¿Por qué la CPU no puede ejecutar un programa desde disco?, la respuesta es fácil de entender si volvemos a nuestras explicaciones anteriores.

Como ya dije, la CPU es muy ordenada. En su pizarra IP siempre lleva el control de que instrucciones van a ser ejecutadas en cada momento. Necesita este número para que cuando ejecuta un FETCHING, no tenga más que indicar que quiere la instrucción situada en la posición X de todas las que componen el programa. Lo que ocurre es que, como estudiaremos más adelante, nuestro programa puede dar saltos adelante y atrás. Por ello, el programa no se puede ejecutar con la suficiente rapidez desde disco.

Suponed que tras ejecutar la instrucción de la posición cero de nuestro programa damos un salto a la posición 23456. Si nuestro programa estuviese en el disco, tardaríamos un tiempo considerable en posicionar las cabezas del disco en la nueva posición de lectura para que se pudiese realizar el necesario FETCHING de dicha instrucción. Como comprenderéis, nuestro programa se haría tan lento, que rompería nuestros nervios rápidamente.

Además, cada vez que hubiese que guardar un resultado, tendríamos que grabarlo en el disco. Cada vez que hubiese que actualizar una variable, cosa que ocurre constantemente, perderíamos un tiempo muy valioso. Para colmo de males, cada posición de programa en el disco tendría que estar etiquetada de alguna manera para que la CPU pudiese dar con ella. En resumidas cuentas es algo imposible.

Esta locura que acabo de describir, no se puede ni soñar en informática, cuya principal máxima es la rapidez.

Por ello, los diseñadores del PC buscaron un banco de trabajo más asequible y veloz; La memoria RAM. Esta, es más rápida de acceder y por lo tanto los tiempos del FETCHING, se ven drásticamente reducidos. Por otro lado, implementando un chip especial para el acceso a los bancos de RAM, la CPU tan sólo tiene que pedir la instrucción que está situada en una determinada posición por el número de ésta y con una velocidad increíble, y casi impensable, dispone de ella para decodificarla.

Podemos pensar en la memoria como en una enorme fila de cajas que están alineadas. Cada caja tiene una etiqueta que la hace única. Por eso el microprocesador puede hacer mención a una de ellas sin riesgos a confundirse en este tratamiento.

Las cajas son como las de botellas de cerveza, lo malo es que en cada una de ellas sólo caben ocho. Podemos, pues, tener 0 botellas, 1 botella, 2 botellas, 3 botellas, ..., 8 botellas. Si disponemos de 1000 cajas y en cada una de ellas hay espacio para 8 botellas, dispondremos de 8000 posibles botellas. Esta misma regla se aplica si tenemos un millón de cajas.

## El Sr. Boole

El Sr. Boole fue un matemático muy inteligente que dio las bases, sin saberlo, de lo que hoy en día son los ordenadores. Sin este amable científico, os puedo asegurar, nuestro PC no hubiese visto la luz jamás.

Hoy por hoy, todo el mundo sabe pensar rápidamente en lo que a los números se refiere. Esto es gracias al sistema decimal de numeración que usamos normalmente.

Sabemos que si este mes cobramos cien mil pesetas y tenemos que pagar el colegio de los niños, que son 40.000 pts, sólo nos quedan 60.000 para irnos de juerga.

Estos cálculos los hacemos con gran velocidad, porque desde pequeños estamos acostumbrados. Pero, por desgracia, a nuestro PC lo mandaron a una escuela diferente.

Es sistema decimal cuenta con 10 números del 0 al 9. Este es también llamado "en base 10".

Bueno, y con tan pocos números, ¿Cómo podemos representar todos que queramos?. Muy sencillo, lo que en realidad da valor a una cifra es la posición que esta ocupa dentro de un número. Veamos este ejemplo:

El número 13526 es diferente a 36521 y, sin embargo, tienen los mismas cifras: 1, 2, 3, 5 y 6. Lo que ocurre es que el 3 está colocado en una posición diferente en cada número y por lo tanto tiene diferente valor.

¿Cuál es su valor?, el valor es siempre 10 veces mayor cuanto más a la derecha esté situado:

El número 13256 en realidad se obtiene de la siguiente operación:

```
1x10000
3x1000
5 x 100
2x10
6
--------------- +
13256
```

El 3, en el primer número "vale" 3000 mientras que en el segundo vale 30000, o sea 10 veces más por estar colocado un lugar más a la derecha.

Como vemos, al contar sólo con diez dígitos, tenemos que hacer un pequeño truco para poder generar todos los números posibles, desde el cero hasta el infinito.

## La escuela del PC

Como comenté antes, a nuestro ordenador lo enviaron a una escuela diferente. El Sr. Boole que era el profesor del colegio, se llevó varios días pensando en como podría enseñar matemáticas a sus alumnos, que eran tan torpes que tan sólo eran capaces de trabajar con dos números; los unos y los ceros. Pensó en el sistema decimal, que, como vimos, cuenta con diez diferentes, y empezó a darle vueltas al asunto. Entonces descubrió que los unos y los ceros también podían adquirir valores diferentes, dependiendo de donde se situasen dentro de la cifra. Por lo pronto, y dado que sólo contaba con dos cifras diferentes para construir sus números, lo llamó sistema binario.

Un número binario se forma al igual que uno decimal, es decir, mediante una consecución de dígitos ordenados de derecha a izquierda. Los números situados más a la izquierda poseen más valor que aquellos que se localizan en posiciones más a la derecha.

De esta forma, el número binario,... un momento. Para no estar todo el rato especificando si se trata de un número binario o decimal, a partir de ahora le voy a añadir una **b** o una **d** detrás.

Como decía, el número 0001b tiene un valor diferente a 0010b y sin embargo, al igual que en el ejemplo de los números decimales, existen las mismas cifras en ambos; tres 0 y un 1, pero, eso sí, colocados en diferente posición. Como siempre los que pertenecemos a otra escuela, los humanos, tenemos que traducirlos a nuestro idioma para poder entenderlos. Para ello se usa la siguiente regla (La posición del BIT dentro del BYTE, como hemos visto indica su valor, los valores de cada BIT se suman para acceder al número final):

| Posición del bit | Valor |
|---|---|
| 0 | 1 |
| 1 | 2 |
| 2 | 4 |
| 3 | 8 |
| 4 | 16 |
| 5 | 32 |
| 6 | 64 |
| 7 | 128 |

Por ello, por ejemplo, el número 01001001b corresponde a 73d y 10000011b a 131d.

Cuando los alumnos de la clase del profesor Boole, nuestros PC's, aprenden el sistema binario se dan cuenta de que ellos trabajan sólo con números de 8 dígitos llamados bytes. Cada byte tiene, pues, 8 bits.

Según esto, sólo se pueden representar los números binarios comprendidos entre 00000000b y 11111111b, o sea 0d y 255d.

Cuando el microprocesador mira los números siempre ve ceros y unos, nunca números decimales como los entendemos los humanos. Esto se debe a que él no tiene cerebro. Es sólo un chip electrónico cuya "inteligencia" es fruto de una serie de corrientes eléctricas (unos 5 voltios) que existen (uno) o no existen (cero).

La memoria RAM, por ejemplo, también se construye con esta filosofía en mente. Cada celdilla de memoria tiene 8 conmutadores que guardan los estados eléctricos de cero y uno mientras estén alimentados por la corriente de la red.

Dentro del ordenador todo tiene el adjetivo mágico de los múltiplos del 8. El BIT (Binary digiT=dígito binario) es la mínima expresión con significado dentro de la matemática binaria.

Resumiendo, el ordenador trabaja con un idioma numérico diferente al nuestro. Ellos usan cifras binarias, que se construyen sólo con unos y ceros y nosotros entendemos los decimales, que tienen hasta diez dígitos diferentes. El ordenador siempre trabaja con 8 bits, un byte, pudiéndose representar hasta el número decimal 255 como máximo. Se hace necesaria una traducción entre ambos sistemas numéricos, porque los humanos no somos capaces de entender los números binarios directamente.

## Un ordenador en un CHIP

Esta archiconocida frase, resume una gran verdad sobre los microprocesadores, porque es cierto que, con el transcurso del tiempo, se les ha ido dando muchas de las funciones que antiguamente tenían asignadas otros componentes del hardware del ordenador. Esta potentísima pastilla de silicio se ha ido apoderando, progresivamente, de las funciones de proceso (ejecución del software) y de las funciones de control (supervisión de hardware), por lo que se dice de ellos que son ordenadores de una sola pieza.

Desde que la informática moderna ha ido desarrollandose en nuestros días hasta límites insospechados, llegando como cualquier otro electrodoméstico a nuestros hogares, el hombre se ha dedicado a pulir, cortar y mejorar el microprocesador como si de un gran y preciado diamante se tratase. Se han conseguido logros asombrosos en este campo, pero todavía no se ha tocado techo, ni muchísimo menos.

Los microprocesadores modernos deben su tecnología a dos descubrimientos o innovaciones fundamentales de la era contemporánea. Por un lado la ciencia de los semiconductores. Por otro, la aplicación de ésta a el campo de los transistores, pilar esencial de la electrónica digital moderna.

El semiconductor, en este caso el silicio con impurezas, tiene la particularidad de dejar que la corriente pase através de él de forma que no presenta la conductividad del cobre ni la resistividad del plástico. Se podría decir que, debido a su naturaleza, se encuentra en un término medio. Cosa que puede parecer una trivialidad a los ojos del profano, pero esta característica especial del semiconductor lo hace especialmente útil en electrónica.

Los materiales derivados de la semiconducción se aplicarón, con gran éxito, en el campo de los transistores. Estos son un conjunto de láminas de metal y silicio de varias clases (Silicio T, N y P) que permiten que una corriente menor pueda controlar otra corriente mayor. Aunque no entendáis esto, no debéis darle más importancia de la que tiene ya que no es el motivo de nuestro estudio. Se hace referencia a ello, simplemente para que os hagáis una idea de como funciona todo.

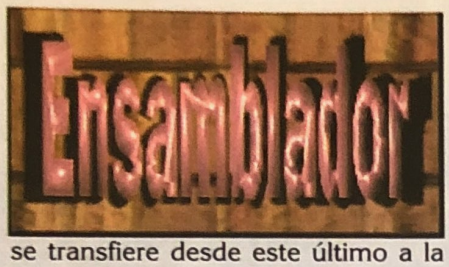
En definitiva, y para que todos lo entendamos, el transitor permite recordar estados eléctricos, es decir, los estados de corriente SI y corriente NO, o ABIERTO o CERRADO, que es lo que popularmente se conoce como 1 y 0. Gracias a ello, y de acuerdo con el Sr. Boole se pueden realizar todas las operaciones de almacen (memoria) y cálculo que el ordenador desee. La combinación de transistores permite, también, formar unas estructuras especiales denomianadas puertas lógicas que son la base fundamental de la aritmética Booleana que estudiaremos más adelante.

Así pues, gracias a los transistores podemos gurdar, mientras exista una corriente eléctrica que los alimente, datos en la memoria y realizar las operaciones que sean menester con ellos.

Otros elementos electrónicos son capaces de guardar la información, como por ejemplo las válvulas de vacio (con gas interior) y los relés, lo que ocurre es que no pueden ser tan pequeños como los transitores.

Esto último da pie a un término muy usado en la electrónica digital y en informática; la integración. Por esta se conoce al número de transitores que pueden introducirse en un espacio. A mayor nivel de integración más transistores se pueden meter en una cierta unidad de superficie. El circuito integrado (IC) es el resultado de este proceso constante de miniaturización que también afectó a la CPU, descubriéndose el término de CHIP o trozo en inglés, que es como se conoce a las pastillas de silicio que albergan en su interior componentes electrónicos miniaturizados.

El chip más util es nuestro querido procesador, motivo del estudio de




este mes. La CPU, con niveles de integración conocidos como VLSI o integración en gran escala, llega a tener hasta una decena de millones de transistores por centímetro cuadrado.

### Dentro de la CPU

Mirar dentro y estudiar el comportamiento interno de los componentes del hardware del ordenador, debe ser la principal afición del programador de vídeojuegos y, en su defecto, el que sea aficionado a la programación en ensamblador. Gracias a este hobby de fisgar en todo, lograremos adquirir unos conocimientos que nos permitirán entender el comportamiento de las instrucciones del procesador, los chips de la VGA, el disco duro, los puertos de comunicaciones, etc. Además, muchos de los problemas de entendimiento o comprensión se solucionan gracias a ello. Por eso os recomiendo que estudiéis siempre bien a fondo lo que se conoce como "programación a nivel hardware", ya que si así lo hacéis el ordenador dejará de tener secretos. En mis artículos siempre iremos pasando del bajo nivel al nivel hardware, indistintamente.

En lo que se refiere a la CPU, vamos a empezar una leve pasada por encima para entrar más a fondo a continuación.

En primer lugar vamos a ver las partes del microprocesador. Una de ellas, y de las más importantes, son los registros. Como indicamos en el símil del carpintero Micro, gracias esas pizarras, él podía apuntar los resultados de los cálculos y los datos que necesitaba para las operaciones. Después los borraba éstos y a continuación reutilizaba las pizarras para los siguientes operaciones.

Los registros son un almacenamiento temporal de información. En realidad son como unas cuantas celdillas de memoria RAM, pero dentro del procesador de forma que son más facilmente accesibles y mas rápidamente que si huebiese que usar el BUS para acceder a ellos.

Por ejemplo, si tenemos que mover un número de una posición de memoria a otra primero llevamos éste a uno de los registros y después se transfiere desde este último a la posición de memoria nueva.

Otra parte sería la unidad decodificadora, encargada de recibir los pulsos eléctricos que llegan a la CPU en forma de instrucciones. Como vimos, un programa se compone de órdenes o instrucciones y los datos sobre los que éstas realizan su acción. Las primeras son interpretadas por la unidad de decodificación la cual recoge la instrucción del bus y la interpreta según el juego de instrucciones del procesador para ver si es una orden válida. Por ejemplo, los pulsos digitales 001011000b indican al procesador que va a hacer una resta.

Dentro del procesador, grabado en su hardware, existe un programa, denominado microcódigo, que se encarga de ejecutar la resta, o la operación que sea. Son los pasos a seguir y es el último de los programas del ordenador y el más importante. Dependiendo de este microcódigo, así será el juego de instrucciones. El de la casa INTEL esta creado por ella mismo y por lo tanto patentado por esta compañía.

Cuando se trata del mundo exterior al procesador estamos hablando de otra parte importante del mismo; la unidad controlodora del BUS, que se encarga de todo lo que concierne a la salida y entrada de datos a/desde la CPU.

Además de las partes descritas anteriormente, existen otras no menos importantes como la unidad de control, la ALU o unidad aritmético lógica y la unidad de FETCHING, que iremos estudiando poco a poco.

Lo que si os quiero decir es que el comportamiento interno del procesador es de vital importancia para entender la programación del PC en lenguaje ensamblador, por lo que deberemos prestar bastante atención.

## PARTES A FONDO

### 1. Los registros de la CPU

Gracias a ellos, el procesador tiene una zona de trabajo donde almacenar temporalmente información de control y datos de nuestro programa. (Os recomiendo la lectura de Herramientas los Debuggers, de CJ1 y 2). Estos almacenes temporales, de rápido acceso por parte de la circuitería del microprocesador retienen la información siempre que no la cambiemos con una nueva información de entrada.

Algunos de ellos son competencia directa de la propia CPU y otros pueden ser cambiados siempre que queramos mediante nuestro programa. Esto que acabo de explicar es importante, porque no todos los registros son accesibles directamente, ya que es el procesador el único que puede cambiarlos.

Para su identificación lógica reciben un nombre de los letras que nos permitirá hacer referencia a ellos desde nuestro programa ensamblador.

Los registros se pueden dividir en las siguientes categorías, en función de la utilización especial que les de el procesador:

1) **Registros de datos de 16 bits**: AX, BX, CX, DX
   Entre ellos distinguimos
   - El Acumulador AX
   - El registro Base BX
   - El registro contador CX
2) **Registros de Segmento de 16 bits**
   - Segmento de DATOS: DS
   - Segmento de CODIGO: CS
   - Segmento de PILA: SS
   - Segmento EXTRA : ES
3) **Registros de PILA**
   - SP, o puntero de pila
   - BP, Puntero Base
4) **Registros índice**
   - SI, índice Fuente
   - DI, índice Destino
5) **Registro de Banderas, FLAGS.**
6) **El puntero de instrucción, IP**

Todos los registros pueden ser utilizados para nuestros intereses dentro los programas, salvo el registro IP que es controlado directamente por el procesador y sobre el que tenemos poco o ningún control.

---

Y con esto ponemos punto y final, por este mes. En Agosto seguiremos con el estudio de los registros y haremos nuestro primer programa en ensamblador.

# TU TIENDA DE INFORMATICA SE LLAMA

HAZ TU PEDIDO POR CORREOS, TELEFONO O FAX (24 HORAS)

TELEFONO Y FAX: 956-813129

APARTADO DE CORREOS: 114

CODIGO POSTAL 11520, ROTA CADIZ

**TARJETA MODEM FAX INTERNA BITFAX Y BITCOM PARA WINDOWS EN CASTELLANO**

**18500 PTS**

**SCANNER DE MANO GENIUS 4500A CON OCR PARA WINDOWS**

**18500 PTS**

**VIDEOCONSOLA SEGA MEGADRIVE CON 7 JUEGOS Y MENACER CON 6 JUEGOS TODO POR SOLO**

**34200 PTS**

**FAX**

**APELSOUND FA 350 envio, recepccion, copia**

**43500 PTS**

**OFERTA ESPECIAL SI ERES SUBSCRIPTOR DE CODIGO JUEGO GOZARAS DE UN DESCUENTO ESPECIAL.**

IVA INCLUIDO

GASTOS DE ENVIO NO INCLUIDOS

PAGO POR TRANSFERENCIA O CONTRA-REEMBOLSO

# SUPER OFERTA DE LANZAMIENTO

ORDENADOR INTEL 486 DX2 66
8 MB DE MEMORIA RAM
S.V.G.A. LOCAL BUS IMB.
CONTROLADORA DISCO EN BUS LOCAL
DISCO DURO DE 340 MB.
CD-ROM SONY INTERNO
(MULTI SESION PHOTO-CD. DOBLE VELOCIDAD.
COMPATIBLE CD-I. CD-AUDIO)
TARJETA DE SONIDO SOUND BLASTER PRO
MONITOR COLOR 1024/768 0.28
FORMATO ORDENADOR MINITORRE
TECLADO EXPANDIDO
RATON 3 BOTONES
MS-DOS 6.2 WINDOWS EN GRUPO 3.11
ORIGINALES DISCOS MANUALES
TARJETA DE REGISTRO
2 JUEGOS CD-ROM ENCICLOPEDIA MULTIMEDIA

## 319.130 PTAS.

I.V.A. APARTE

PORTES EN PENINSULA PAGADOS

# Carta del Lector

## LA CULPA FUE DE...LAS HORMIGAS

**Nuestro amigo Antonio, encontró su afición a la informática de mano de un juego. A cualquiera le puede ocurrir algo así. Si tienes una historia parecida que quieras compartir o algún tipo de suceso que nos quieras relatar, no lo dudes, mándanosla a:**

**Código Juego**
**Ajatel S.L.**
**Apartado 114**
**11520 Rota**
**(Cádiz)**

Cuando todo esto de los juegos de ordenador comenzó, hace ya más de diez años, no me lo podía creer. Concretamente, recuerdo un juego para el ZX Spectrum que me dejó atónito. Creo que fue en ese mismo instante, cuando me di cuenta de lo que podía llegar a representar un ordenador para el área de entretenimiento y el ocio.

El programa del que hablo, se llamaba algo así como HORMIGAS (ANTS). Era el primer juego, que yo había visto, que representaba un mundo tridimensional con tanta realidad, y eso que, por las escasas prestaciones del Spectrum, el juego era en blanco y negro, pero os aseguro que este detalle no se notaba. El caso es que aquellas hormigas asesinas perseguían a nuestro personaje estrella (hombre o mujer) implacablemente por todo aquel maravilloso escenario de edificios formados por bloques de piedra.

Tal fue el shock que sufrí ante tan buen juego, que inmediatamente decidí estudiar un poco el programa por dentro para averiguar como aquel programador inglés pudo hacer semejente obra de arte en un ordenador tan lento y pobre como el ZX Spectrum. Rápidamente me vi rodeado de todo tipo de listados y, haciendo honor a la verdad, no lograba entender nada de nada. Tanto es así, que acabé aburriendome y abandoné el proyecto. Le devolví la impresora térmica a mi amigo y me dediqué a otros meneste-res.

Algún tiempo más tarde, en un viaje que hice a Córdoba, entré en una librería enorme de esta bella y antigua ciudad. Buscaba un diccionario de antónimos y sinónimos cuando, de repente, en una de aquellas polvorientas repisas pude leer el título de un libro: "Programación ensamblador Z80". Lo abrí, le eché un vistazo y me dije a mí mismo que por qué no. Me gasté todos mis ahorros en aquél esplendido manual y en otro libro que compré al día siguiente "Las 40 mejores subrutinas para el ZX Spectrum".

Armado de paciencia, empezó mi aventura en el código máquina. Poco después ya hacia rutinas que dibujaban y movían sprites por la extraña pantalla del ZX Spectrum. Finalmente, meses más tarde, retomé el listado del juego de las hormigas y pude aprender muchas más cosas. Poco a poco, me fui enganchando cada vez más hasta acabar totalmente colgado por la programación en ensamblador.

En la actualidad, realizo todo tipo de programas en distintos tipos de ensambladores 80x86 y Motorola 68000, y ya el Z80 quedó para la historia. Estoy considerado como un buen programador entre mis amigos, los cuales, al no trabajar a bajo nivel con el ordenador, por programar en CLIPPER y otros lenguajes, me bombardean con preguntas de todo tipo sobre las interioridades del PC. Jamás me arrepentiré de haber aprendido el lenguaje ensamblador, por la potencia de programación que ahora soy capaz de desarrollar en mis aplicaciones, como el compresor de datos que estoy creando y que es capaz de comprimir incluso más que PKZIP.

Me alegro de que haya salido una revista como la vuestra al mercado y de que se instruya a los programadores a ser más conocedores del ordenador que manejan.

Gracias Código Juego, os deseo lo mejor.

**Juan Aguilar Sempere**
**Madrid**

# La tintilla informática

## La decisión de Sophie

✐ **Miguel A. Lobato**

Hola amigos! De nuevo con todos ustedes. Desde aquí, quisieramos daros las gracias por adquiridnos en el quiosco. Apelamos a vuestra benevolencia escritora y lectora, por permitirnos dejar aparcadas, por el momento, algunas quejas recibidas que serían interesantes comentarlas, pero el espacio es limitado. Somos conscientes y no nos olvidaremos de ellas. De todas las recibidas, hemos destacado la de este amigo por el trasfondo que tiene.

Sin más dilación, vamos a la queja de este número.

### EXPOSICIÓN DE LOS HECHOS

Francisco Arjona, nos escribe desde Roquetas de Mar (Almería) para exponer, lo que a su juicio es un atraco a mano armada en cuanto a tiempo y dinero supone esta cosa de la informática; me permito decir "Cosa Nostra" (siguiendo con el símil), pues son muchos los llamados, pero pocos los elegidos quienes la disfrutan a plena potencia.

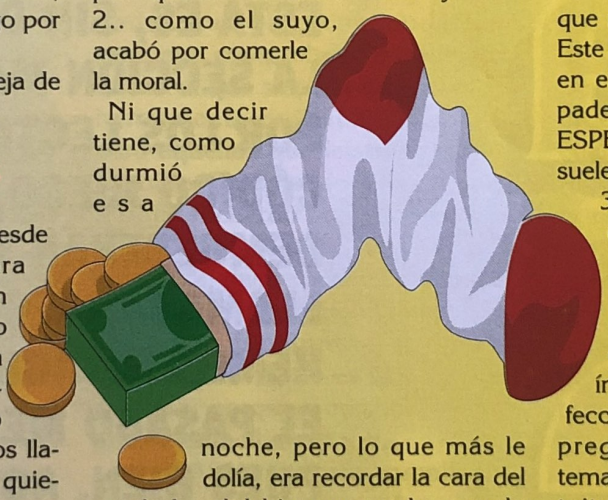
Paco, empezó con la informática hace ahora dos años. Su padre, único poseedor del dinero plástico y del sonante también, ante la insistencia y unas notas dignas de admiración, no tuvo más remedio que acceder a las pretensiones de su hijo. Le compró su primer ordenador personal, igualito al de su más íntimo amigo Jorge, que destrozaba tres teclados por minuto de tanto jugar a los marcianitos.

¿Cuál pensáis que era?, un magnífico y maravilloso 286 ¡todo un lujazo! Eso sí, siempre bajo la atenta mirada del vendedor del hipermercado de turno, que no escatimó en esfuerzos, colgándole no sólo el ordenador, sino por supuesto, una impresora de igual marca, ¡ya podréis imaginar cual!

Poco tiempo después, Paco se percató al ir a casa de Ernesto, que su poderosísima máquina nueva, no se ajustaba del todo a sus necesidades. Los colores no eran nítidos y ¡Diantres!, el juego del superespía corría a la velocidad de su hermana anoréxica cuando su madre decía ¡A comer!

Al preguntar a Ernesto, éste le soltó aquello que dice: el 286 está acabado, el mío es un 386 a 25 MHz.. Lo de los 25 megaherzios, le sonaron a pegote, pero que Ernesto tuviera un 3.. y no un 2.. como el suyo, acabó por comerle la moral.

Ni que decir tiene, como durmió esa noche, pero lo que más le dolía, era recordar la cara del vendedor del hipermercado, con la americana cruzada y "fijata" en el pelo diciéndole "Chaval te llevas lo último, lo mejor...."



### COMENTARIO

Me he extendido un poco en describir las andanzas de nuestro amigo Arjona. Extrayendo lo que a mi juicio considero realmente importante: En informática, lo nuevo queda rápidamente antiguo, y no por ello inservible. Imagino que en algún momento todos hemos pasado por una situación más o menos similar.

Francisco, desde estas líneas, se me ocurren varias cosas que tanto a tí como al resto de los usuarios de material informático les valdrán de gran ayuda. Algunas os podrán resultar obvias, pero no por ello, deberíamos menospreciarlas.

1.- Huye como alma que lleva el Diablo de comprar cualquier producto informático en los grandes almacenes NO ESPECIALIZADOS. Su misión es vender a cualquier precio, da igual que sean ordenadores que latas de tomates fritos. No te decidas, al pensar que es más barato, pues a la larga te saldrá bastante caro. A las pruebas me remito.

2.- EL APOYO TECNICO es una cosa que se me antoja de vital importancia. Este sólo lo encontrarás en los amigos, en el club... más avanzados que han padecido algo similar o en TIENDAS ESPECIALIZADAS, donde cada cliente suele ser salvo excepciones, un amigo.

3.- AHORRO DE DINERO. Evitar desplazamientos innecesarios al lugar de adquisición (el oro negro queridos amigos) porque se te olvidó preguntar por algo, solicitar consejos prácticos de cualquier índole al vendedor incluso confeccionarse un papel con las dudas o preguntas que tengamos sobre el tema. A buen seguro, te lo darán gratuitamente, por el simple hecho de adquirir el producto, por pequeño que sea, allí mismo.

4.- EL ULTIMO GRITO en informática no lo puede dar nadie. Es una quimera, cada día te encuentras con algo nuevo en el mercado que supera lo que tienes, y no por ello lo tuyo ya no sirve.Trata de sacarle el máximo partido, si realmente ya no puedes más. Compralo.

### MORALEJA

BUSQUE, COMPARE Y SI ENCUENTRAS ALGO MEJOR, PIENSATELO BIEN.

❑❋❑

# Así se hace un juego para PC

ESTA ES, SIN DUDA ALGUNA LA SECCIÓN MAS ESPERADA POR LOS LECTORES DE CÓDIGO JUEGO, Y ASÍ LO DEMUESTRAN VUESTRAS LLAMADAS Y CARTAS QUE HEMOS RECIBIDO DURANTE EL PASADO MES.

PUES BIEN, AQUI TENÉIS OTRA ENTREGA PARA QUE PODÁIS SEGUIR POCO A POCO TODO EL PROCEDIMIENTO DE CREACIÓN DE ESTE SUPERVIDEOJUEGO QUE SERÁ MOOKIE.

EN ESTA OCASIÓN ESTUDIAMOS EL LISTADO DEL PASADO MES, SOBRE LA PRESENTACIÓN EN PANTALLA DE LA INFORMACIÓN GRÁFICA.

Hola a todos de nuevo. Espero que, por lo menos, hayáis echado un vistazo al listado del mes pasado...¿Cómo?, ¿Qué no habéis podido?, bueno, no importa. Lo único que debéis hacer es seguirme en este artículo y pronto podréis hacer vuestra presentación propia.

Son muchos los que han contactado conmigo en referencia a las explicaciones que dábamos en el número anterior de Código Juego. La duda principal era por qué no se había incluido el listado fuente (.ASM) en el interior del disquete que acompañaba la revista. Bien, os diré que por una razón muy sencilla. El programa MOOKIE, irá creciendo un poco todos los meses. Esto quiere decir que cuando añadamos, el mes que viene, una porción de listado más, el programa será el mismo que el que tenemos hasta el momento con esa nueva parte dentro. Por ello, y con la finalidad de no estar repitiendo el mismo listado hemos pensado que más valía esperar a que el programa fuese un poco más grande para que fuese incluido dentro del disquete. De todas formas, para aquellos forofos coleccionistas, el listado del anterior esta incluido en el Superdisco número 1.

Bien, entrando ya en la materia que nos importa, empecemos con el estudio de MAIN.ASM, que es, hasta el momento nuestro programa de juegos.

Como explicamos el mes anterior, todo juego puede dividirse en una serie de partes lógicas, que no tienen porque ser fijas, pero que podemos encontrar en la gran mayoría de los juegos para ordenador contemporáneos.

La primera en orden, y no por ello la más importante, es la presentación. Mediante ésta, introdujimos a la persona que va a disfrutar del juego, sobre que trata éste. Además podemos introducir una serie de menús de opciones y tablas de información que no quedarían bien dentro del propio contexto del juego.

Vimos, también, que ya que se trata de sacar información por la pantalla, era mejor hacerlo de forma elegante. La mejor de todas la maneras es causar impacto en el que lo ve y por ello acudíamos a trucos como el *fundido*, que permite hacer efectos de aparición de la información de forma "diferente". Pues bien ahí no termina la cosa. Se puede realizar todo lo que queramos , como poner mensajes que se mueve al aparecer en pantalla, scroll de fondo o de los propios mensajes, rotación de la información, etc.

Por ahora nos centramos en el fundido y exactamente en el tipo que hemos utilizado para MOOKIE.

## MAIN.ASM

El programa o listado está creado utilizando el MACROASSEMBLER de MICROSOFT. En esta misma revista se ha incluido un artículo sobre sus características y posibilidades que os recomiendo leáis para que comprobéis, por vosotros mismos, que se trata de una superherramienta de trabajo. También, para aquellos que leísteis el artículo sobre "Trabajando a dos monitores" os recuerdo que con la utilización de CodeView, mediante la opción /2, desde la línea de comando, estaréis en mejor situación de ver como funciona el programa paso a paso. Si no es así, no preocuparse, porque por ahora el programa es fácil de seguir.

MAIN.ASM, una vez compilado y cuando lo hacemos funcionar desde el disco duro, puede resultar simple. Quiero decir con esto que más de uno habrá pensado que el programa hace poco por ahora y la verdad es que sí. Sin embargo, en MAIN.ASM hay muchas cosas importantes que todos debemos aprender. Estas son:

1) Entrada, comprobación e inicialización del sistema
2) Cuerpo principal del programa, en este caso es el fundido.
3) Subrutinas de apoyo.
4) Situación del sistema en modo original.
5) Salida al sistema operativo.

Estas partes son siempre, o deben serlo, comunes a todos los programas. Pasemos al estudio detallado de cada una de ellas.

## ENTRADA E INICIALIZACION

Para que un programa arranque, teniendo como anfitrión al sistema operativo MSDOS, se produce un proceso que permite que éste sea incluido en la memoria y que encuentre a su disposición todos los recursos del sistema.

El hecho de que cualquier programa que sea activado desde la línea de comandos, o desde dentro de otro programa, se haga el dueño y señor del ordenador, es una característica intrínseca de este sistema operativo, que como todos sabemos muy bien, no es multitarea, o sea que no permite que varios programas tengan el poder del ordenador al mismo tiempo. Por ello, cuando se activa cualquier aplicación, el MSDOS le da toda la memoria de que dispone. Esto implica que si nuestro programa, ya sea un vídeojuego o cualquier otro, no va a utilizar toda la memoria, tendremos que liberarla o soltarla. A esto se le conoce como ser educados. Es decir, debemos inicializar nuestro programa de forma que "suelte" todo aquello que no va a necesitar realmente. Esta necesidad debe contemplarse especialmente si estamos creado un programa TSR, ya que si no liberamos la memoria, ningún otro programa podrá ser cargado sin que el DOS nos diga enfadado "NO HAY MEMORIA SUFICIENTE".

Si no deseamos que ningún otro programa coexista con el nuestro, tan sólo deberemos ser "maleducados" y hacernos dueños de todo. En el caso de MOOKIE vamos a ser corteses por propia conveniencia. Liberaremos la memoria para que siempre sepamos si tenemos memoria RAM suficiente para seguir trabajando. Más adelante veremos cómo.

Cuando escribimos MOOKIE desde la línea de comandos, o sea desde el prompt del DOS, y pulsamos ENTER se desencadena una serie de acontecimientos que, como decíamos, permiten que

cualquier programa pueda ser ejecutado desde el MSDOS. El interprete de comandos COMMAND.COM se encarga de analizar nuestra orden y de buscar en el directorio actual y en aquellos que estén incluidos en el PATH, un archivo con ese nombre. Cuando lo encuentra comprueba si es ejecutable (.EXE, .COM o .BAT), y en caso contrario nos dice "COMANDO O NOMBRE DE ARCHIVO INCORRECTO".

La parte del MSDOS que permite la ejecución de un programa se llama función EXEC. Esta, construye una tabla de datos especial en la memoria RAM, llamada Prefijo de Segmento de Programa o PSP, a partir del cual introduce los datos del propio programa que son cargados desde disco. A partir de aquí, realiza un importantísimo procedimiento de Reubicación, o adaptación de los saltos y posición de las variables del programa a la situación exacta donde ha sido introducido en memoria. La reubicación es necesaria ya que el microprocesador divide, o segmenta, esta última de una forma especial. Esto impide que un programa se ejecute correctamente en dos posiciones de memoria diferentes si no se realiza la reubicación.

Una vez que el programa se carga y se reubica en memoria el sistema operativo le da el control. El salto de entrada en nuestro programa se produce a la dirección de inicio que está guardada en la cabecera de programa (en el caso de un programa con el formato .EXE) o en la posición 0100h del segmento de memoria donde haya sido introducido (en el caso de un programa con el formato .COM). Esto se estudiará, en profundidad, en el curso de ensamblador. Aquí ya presuponemos que domináis estos conocimientos.

En el caso de MAIN.ASM, el punto de entrada está en la sentencia etiquetada con MOOKIE, justo al comienzo del listado.

Así que ya tenemos el programa metido en memoria, reubicado y funcionando justo al principio.

La primera orden es CLD que nos asegura que la dirección de las transferencias de datos con ordenes como REP STOSB serán siempre en orden ascendente de dirección de memoria.

A continuación ajustamos el valor del segmento de datos. Cuando entramos en un programa del tipo .EXE, que es el caso de MOOKIE.EXE, el valor del segmento de datos es siempre igual al del segmento de código. Cómo todas las variables de nuestro programa hemos decidido introducirlas en un segmento diferente, para que podamos acceder a ellas se hace imprescindible retocar este valor. Para ello, cargamos en AX el valor del SEG DATA y lo introdujimos en DS con MOV DS,AX. Te recuerdo que la orden MOV DS,SEG DATA no existe, por lo que lo hacemos en dos pasos, con el registro AX como almacén temporal. También guardamos el comienzo del PSP del que hablábamos antes, porque nos va a hacer falta más tarde. El registro ES contiene ese valor a la entrada de un programa.

Seguidamente hacemos una comprobación muy importante; la versión del MSDOS. Como sabemos, el sistema operativo siempre queda en segundo plano cuando una aplicación, diferente a él, esta corriendo. Las funciones que están disponibles gracias a ello ahorrarán mucho trabajo al programador de los juegos, que puede utilizarlas a su antojo y siempre que encuentre en ellas un medio de apoyo. Sin embargo, cada versión ascendente del DOS ha incluido funciones nuevas y mejoras de las anteriores. Conscientes de ello, tendremos que comprobar que versión del DOS está funcionando en el ordenador donde se ejecute MOOKIE en ese momento. De esta forma sabremos y nos aseguramos de que no encontraremos problemas posteriores durante el funcionamiento del juego. Desgraciadamente, deberemos impedir a todos aquellos que posean una versión inferior del sistema operativo de la que se requiere en nuestro juego a que puedan jugar con él, mediante el error "versión incorrecta de DOS". En nuestro caso hace falta la versión 3.20 o superior.

Estamos en la etiqueta DOS_OK. En esta parte utilizamos la función o interrupción 21H (Léase "El MSDOS y el programador de vídeojuegos") subfunción 35h para averiguar cual es el comienzo de la subrutina de servicio de la interrupción 09h. Esta última se encarga de gestionar el teclado del PC. Cada vez que pulsamos una tecla, se produce una interrupción hardware que hace que se salte a esta rutina para la gestión de la tecla pulsada. En MOOKIE vamos a crear un subrutina de interrupción especial, diferente a la que nos ofrece el sistema operativo. Por ello, averiguamos donde comienza la del sistema operativo y guardamos este valor para más adelante, cuando se produzca la salida al final al sistema operativo. Como veremos reestableceremos el valor del dato que aquí guardamos en memoria justo en la variable etiquetada como VECTOR_09. Si no cambiásemos el valor al salir el MSDOS se perdería y nos quedaríamos colgados y sin teclado.

Seguidamente hacemos lo mismo con la interrupción que gestiona el temporizador del PC (int 1CH). Como nos vamos a crear nuestro propio reloj de eventos guardamos el valor de inicio de la rutina encargada de ésto para cuando abandonemos el juego.

Acto seguido introducimos los comienzos de nuestros dos gestores de interrupción 09H y 1CH.

Otro parámetro que deberemos cambiar para nuestro juego es el modo de vídeo. Por lo general, el MSDOS arranca en modo 03h (80 x 25 columnas, 16 colores). Nuestro MOOKIE necesitará la plena potencia del modo 13h (256 colores) para poder moverse por su mundo, así que guardamos el modo de vídeo actual en la variable VIDEOMODE (cualquiera que sea su valor) para poder restaurar el modo al salir del juego, tras lo cual, mediante la interrupción 10h (del BIOS de la tarjeta VGA) entramos en el modo de vídeo correcto (MOV AX,13h).

Nuestra pequeña comprobación e inicialización queda así completa. En próximos listados añadiremos

más chequeos que son necesarios al entrar en el programa, para que podamos hacer una salida correcta y educada al MSDOS.

## CUERPO PRINCIPAL

En esta parte es donde se va a ir incluyendo el verdadero núcleo del programa aunque, por ahora, sólo consta de las llamadas a las subrutinas de apoyo que permiten realizar el fundido, motivo de nuestro estudio.

La tarjeta VGA permite programar sus registros internos de forma que podemos acceder a la paleta de colores de manera arbitraria. En el modo 13h, disponemos de 256 registros de color, lo cual quiere decir que a cualquier píxel que encendamos en la pantalla, sólo le podremos atribuir uno de estos colores y no más.

Supongamos que ponemos un único píxel en una posición de pantalla al azar, por ejemplo, la esquina superior izquierda. La posición de memoria de vídeo donde deberemos introducir el dato es justo la primera asignada a la memoria de vídeo VRAM, en el espacio direccionable por el microprocesador:

A000:0000

Los 256 colores posibles se nombran o numeran del 0 al 255. Si introdujimos el valor 123 (7Bh) en esta posición, le indicamos a la tarjeta de vídeo que queremos que el píxel de esa posición apareza con el color 123. La VGA tiene unos registros internos que siempre "recuerdan" los 3 componentes de color (Rojo, Verde, Azul) de ese color en particular. Si estos 3 matices son, por ejemplo, ROJO=255 VERDE=255 y AZUL=255, el color 123 será el blanco y por tanto nuestro puntito de pantalla aparecerá con dicha característica. Si cambiamos el dato por otro aparecerá otro color diferente, siempre que los componentes de color sean diferentes, ya que el color nuevo también puede tener los mismos que el color 123. De esta manera podríamos tener 256 colores iguales si todos los componentes RGB estos últimos fuesen los mismos, cosa que nadie hace sin una buena razón. Pero que creéis que ocurriría si tocamos el RGB de nuestro color 123. Sencillamente que el color cambiaría en la pantalla aun siendo el mismo número de color.

De una sencilla regla matemática sacamos que si tenemos 256 registros de color y cada uno puede tener 3 componentes (RGB) y cada uno de ellos puede variar de 0 a 255, el número total de colores disponibles en bloques o paletas de 256 diferentes sería se 262.144.

En nuestro ejemplo, el color blanco puro tiene, como hemos dicho, los siguientes matices RGB:

Rojo=255
Verde=255
Azul=255

Al acceder a los registros VGA que controlan el color 123 y variar el componente VERDE cambia de la manera que veis en la figura adjunta.

Como podéis comprobar el color se convierte en púrpura y se va apagando en luminosidad sucesivamente. Según el modelo de color RGB, a menor cantidad de matiz más oscuro y a mayor cantidad más cercano estaremos del blanco. Por ello, el negro se logra cuando no hay ningún componente de color, es decir:

Rojo=0
Verde=0
Azul=0

El truco del fundido se aprovecha de esta característica. Los colores originales de un gráfico en pantalla, sea cual sea el contenido RGB de cada uno de ellos, se van oscureciendo paulatinamente en el tiempo de forma que después de varios bucles llegamos al color negro total en cada uno de ellos.

Para lograr esto, y dado que cambiamos los colores, guardaremos la paleta de color en un buffer de memoria especial y desde ahí tocaremos a nuestro antojo los valores de los registros de la VGA.

## CARGANDO QUE ES GERUNDIO

Antes de seguir vamos a ver como se crea la pantalla en cuestión y como se guarda en el formato adecuado para la carga en memoria desde disco.

Los formatos gráficos existentes, PCX, GIF, TIF, TGA, etc, utilizan sistemas de compresión que son muy útiles para ahorrar espacio en disco. Lo que ocurre es que para la explicación del listado he preferido no utilizarlos y grabar las pantallas en formato "crudo", es decir, sin ningún tipo de compresión.

Por eso utilicé una utilidad de shareware que podéis encontrar en el superdisco número 2. Se llama VGACAP y permite capturar una pantalla de gráficos, en modo 320x200 y guardarla como un archivo .SCR. Este archivo tiene 2 partes:

1) 768 bytes, componentes RGB de los 256 registros de color de la VGA en el momento de la captura de la pantalla.

2) 64000 bytes de los pixels. (320x200=64000)

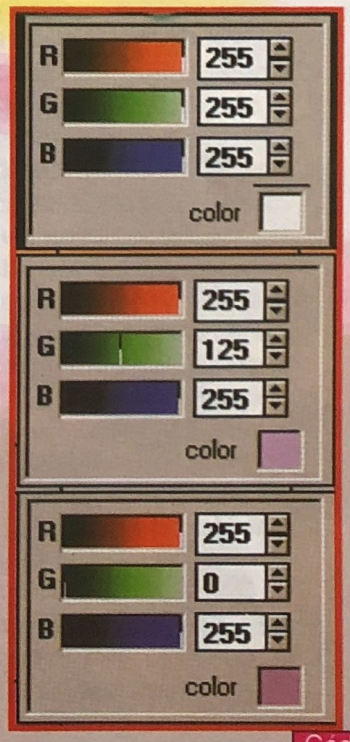
Luego siempre medirá 64768 bytes. Una vez guardado podemos renombrarlo como nos plazca y utilizarlo en nuestros juegos.

VGACAP, sin embargo, es SHAREWARE y deberéis abonar el precio que exige su autor tras un periodo prudencial de prueba.

En esta presentación he usado 8 pantallas diferentes como ejemplo del efecto que queremos explicar. Todas están en formato .SCR, con la paleta de colores al principio y los bytes de pixels a continuación.

Os debo dejar ya. En Código Juego 3 me seguiréis soportando otro poco cuando expliquemos más sobre la carga de archivos usando las funciones handle del MSDOS y el final del efecto de fundidos.

R.L.



Hablamos con los programadores de:

# IGOR: OBJETIVO UIKOKAHONIA

## Péndulo Studios

Yo, como muchos otros, soy de los que se quedan absortos durante horas con un buen juego de ordenador. Por eso, cuando oigo o leo que algo nuevo sale al mercado, espero con avidez a que esté pronto adornando la repisa de la tienda de informática en la esquina de mi calle. Pero cuando averiguo que ese juego reciente es una aventura gráfica, entonces mi interés crece por momentos y me muerdo las uñas impaciente por verlo. Lo que no puedo soportar por más tiempo es cuando descubro, que el juego está hecho por españoles y que su calidad es equiparable a la de cualquier otra vídeoaventura de allende de nuestras fronteras.

Esto fue exactamente lo que me ocurrió con IGOR: OBJETIVO UIKOKAHONIA. La verdad es que no me pude aguantar más y finalmente me decidí a llamarlos. Quería saber más cosas de este grupo de programadores que acaba de lanzar la primera aventura gráfica española. Tenía que saber más cosas sobre lo que sienten y padecen unos jóvenes que, un buen día, deciden dejar todas sus demás aficiones para dedicarse por entero a crear un juego de tan alta calidad, con el que todos no vamos a divertir.

Hable con Felipe Gómez, uno de los integrantes de Péndulo Studios, los verdaderos protagonistas de esta historia.

CODIGO JUEGO: Ante todo, Felipe, hay algo que me intriga, ¿Cómo fue que escogísteis un nombre como ese para un juego?

FELIPE GOMEZ: "El nombre nos sonó desde un principio y aunque después nos rondó por la cabeza algún otro, cuando tuvimos que presentar la demo del juego y el tiempo se nos echo encima decidimos, finalmente, quedarnos con UIKOKAHONIA. Incluso nuestra distribuidora nos apoyó en este respecto, pero creo que la gente lo recordará como IGOR simplemente".

CJ: El nombre de Péndulo, ¿Surge de la misma forma?

FG: Se puede decir que sí. Como todo lo demás fue escogido por todos nosotros y que quieres, después de mucho repetirlo te vas quedando con él, hasta que, finalmente, es algo tan natural que se queda y te suena de lo más normal. No tiene ningún significado especial".

CJ: Los que programamos y creo que todos los que disfrutan con los juegos de ordenador, tenemos una pregunta que sería imperdonable saltarnos, ¿En que lenguaje está programado IGOR?

FG: IGOR está escrito en PASCAL en aquellas partes del programa en las que nos hace falta control y en ENSAMBLADOR en aquellas otras en las que se necesita más alta velocidad de proceso. Es, pues, una mezcla de ambos".

CJ: Supongo, Felipe, que el programa tendrá una estructura interna bien definida, ¿Podrías contarnos algo sobre esto?

FG: El juego consta de un núcleo principal, o interprete, que es reutilizable, es decir, que nos permitiría crear otra aventura con un menor esfuerzo, ya que esta sección de código es la que controla todas las funciones principales de la aventura; salida de diálogos en pantalla, movimiento de los personajes, animaciones, música, efectos de sonido, etc. Por otro lado, está aquella parte que hace que IGOR sea característico y acorde con el guión que hemos creado".

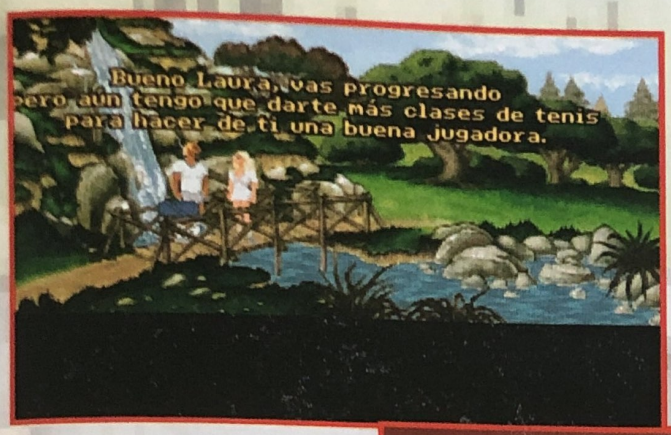
CJ: A propósito, ¿quién hizo el guión?

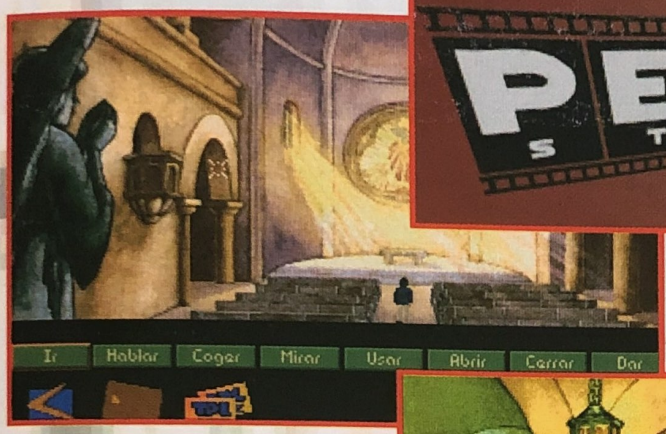
FG: "Lo hemos realizado entre todos, Rafael Latiegui, Ramón Hernáez, Miguel Angel Ramos y yo mismo".

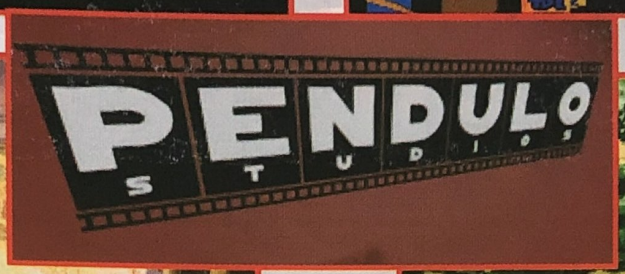
CJ: ¿Cuánto tiempo tardásteis en confeccionar el juego, desde que surge la idea hasta que se termina en su totalidad?

FG: Es difícil decirlo, porque hay un paréntesis en medio. En total sería alrededor de un año y un mes. Al principio comenzamos a ratitos, pero no fue hasta Septiembre del año pasado cuando decidimos imponernos un horario de trabajo, contratar un dibujante y un músico y apretar el acelerador. Gracias a esta última decisión y trabajando muy duro, conseguimos terminar el proyecto".

CJ: En todos los grupos de programación siempre hay alguien





**Distribuido por DRO SOFT**

**Teléfono 91-4291161**

**HD: 10,5 megas**
**T. Gráfica: VGA (256 col.)**
**CPU: 286 o superior**
**Control: Ratón y Teclado**

que es el que une a los demás, ¿quién es el artífice de todo en vuestro grupo?

FG: "Sin duda, Ramón Hernaez. El es el que mejor domina los lenguajes Pascal y Ensamblador y fue el que nos animó para que nos uniéramos a su proyecto. A partir de ahí, todo ha sido un trabajo en grupo".

CJ: "Los gráficos de juego son de una calidad impresionante y digna de mención. ¿Habéis hecho vosotros este trabajo?

FG: "Bueno, los fondos, o sea los escenarios por los que se mueve el personaje, ha sido hechos por Rafael Latiegui y los muñecos y las animaciones de los distintos protagonistas las hemos creado nosotros mismos".

CJ: Lo que sí se nota es que la aventura tiene mucha calidad y que es extensa en contenido y acciones...

FG: "El juego tiene 10 ó 11 megas una vez instalado y la verdad es que

sí que es bastante completo".

CJ: ¿Estáis trabajando ya en algún nuevo proyecto?

FG: "Sí. Ya estamos metidos en otra aventura gráfica que es lo que nos gusta a los que componemos Péndulo Studios.

CJ: Entonces, según me has explicado, ahora os resulta más fácil...

FG: "Efectivamente. Con el núcleo de las aventuras ya programado, sólo tenemos que realizar aquellas implementaciones que representen, en realidad, una innovación en la nueva aventura. Además, por supuesto, hay que encontrar el contexto espacio-temporal adecuado, hacer el guión, los escenarios, etc. También, vamos a mudarnos en breve espacio de tiempo de Arganda del Rey a Madrid capital, luego estamos bastante ocupados últimamente".

CJ: Os deseamos, desde la redacción de la revista que tengáis toda la suerte del mundo, y que nos alegramos profundamente de que un puñado de jóvenes españoles hayan tenido la valentía de acometer una empresa de este tipo.

FG: "Os deseamos lo mismo, ya que vuestra revista también está dando sus primeros pasos en el mundo informático".....

Esta fue la entrevista que mantuve con ellos.

El juego merece la pena de verdad, y no porque queramos hacer apología de lo español, sino porque no tiene nada que envidiar a los que compramos de otras compañías, principalmente del país del dólar.

# El juego del mes

Juego: Sam and Max Hit the road
Traducción: Sam y Max en la carretera
Creadora: Lucas Arts
Distribuidor: ERBE
Hace Falta: 580 Kb de RAM, 2 mb de EMS, 15 mb disco duro

*Cuando un empedernido jugador, como el que escribe estas letras, tiene la oportunidad de encontrarse con una vídeoaventura, como la que hoy comento, uno no sabe si cuando termina de jugar, está apagando el televisor o el ordenador. La sensación es que hemos echado un rato con dos personajes reales de los dibujos animados. Y si no os lo creéis, seguid leyendo.*

Una de las barreras más importantes, con las que nos encontramos los programadores de juegos para ordenador, no es el programa en sí. Con un poco de destreza y aprendizaje, todos los problemas que plantea éste llegan a solucionarse. Lo más traumático es el hecho de no tener una buena idea.

Yo conozco mágnificos programadores que dominan varios lenguajes, y que pueden hacer un programa como el que enciende una cerilla, pero ellos mismos reconocen adolecer de buenas ideas a la hora de crear un videojuego.

Y es que en este mundo de la programación lúdica, ya se han hecho muchas cosas y la gran mayoría, muy buenas.

Por ello, re-sulta di-fícil en-contrar algo original, algo qu rompa con lo común, y cuando nos encontramos un programa como SAM AND MAX, todavía nos queda fé en la imaginación de los creadores de juegos.

## ORIGINALIDAD POR UNTUBO

Sam es un perro gordito y de aspecto bonachón mientras que Max está personalizado por un conejo de gran dentadura y amplia sonrisa. Muchos pensarán que la pareja no pega ni con cola; nada más lejos de la realidad.

A pesar de tener una fisonomía tan diferente y un carácter tan desigual, nuestros dos detectives se compenetran de maravilla, a la hora de resover los numerosos puzzles que nos iremos encontrando por todos los escenarios, imposible contar cuantos, por los que nos vamos a mover.

Lo original a resaltar de esta aventura es, como siempre en los juegos de Lucas Arts, su guión. No hay nada más dispar en estos juegos, que su argumento, como el hecho de tener que rescatar un preciado Yeti que se ha esfumado de un parque de atracciones.

Todo lo que envuelve este genial juego tiene ese toque especial de la saga del SCUMM, como algunos gustan llamarlos.

Las aventuras están llenas de graciosos, irónicos, sarcásticos y bromistas personajes. Los escenarios, son de un sabor carácteristico y de un gusto exquisito. Se nota que detrás de ellos hay un largo número de personas entre dibujantes y diseñadores-.

Cada actor tiene su propia forma de comportarse y lo que más llama la atención son la cantidad de acciones que pueden llegar a realizar.

En definitiva, el argumento, los escenarios, los pesonajes y hasta la música y los efectos sonoros son originales, vamos que no se parecen en nada de nada a los juegos que están en el mercado, ni tan siquiera a los de su propia serie, como Indiana Jones o Maniac Mansion (El día del tentáculo).

Un derroche de originalidad que hace pensar que los señores de Lucas Arts han comprado la varita mágica de la inédito y del buen hacer.

## El SCUMM

Cuando comenzaron a generarse

El interface del SCUMM nos permite realizar todas las acciones del juego con un simple "click"

Como se aprecia en las imágenes, los escenarios son de una calidad impresionante, sumergiandonos de lleno en el argumento de la aventura.

juegos del tipo de aventuras, los ordenadores eran muy pobres en posibilidades, tanto es así que estos juegos no tenían gráficos, sino sólo letras y diálogos.

Recuerdo un juego para el ZX Spectrum que se llamaba "El hobbit". Basado en la popular novela de J.J. Tolkien, el juego era lo más feo que creo haber visto. No es que fuese de mala calidad, es que a lo difícil de tener que escribir todas las ordenes (en inglés), se unía el hecho de que los gráficos eran casi inexistentes.

Para aquella época, resultó un gran éxito, de hecho el que escribe lo jugo hasta el final, pero rápidamente surgió una gran demanda de ese tipo de juegos, pero con más sustancia gráfica.

Algunos años después, nació una compañía que se dedicó por completo a desarrollar un sistema especial de programación que permitiese el uso de gráficos en movimiento y estáticos de forma que el que jugaba pudiese ver lo que ocurría en tiempo real.

Lucas Arts generó, de esta forma, un código que funcionó perfectamente y que enganchó con éxito a los muchísimos partidarios de este tipo de programas.

La idea era hacer que la barrera de tener que escribir las ordenes se suprimiese, a la vez que se disminuía la barrera del idioma, ya que muchas de las acciones eran intuidas por el jugador, sin necesidad de una comprensión mediante mensajes.

De esta forma, frases como "GET SWORD FROM TRUNK", o "COGE ESPADA DEL BAUL" eran cambiadas por un simple click del ratón y "EL PERSONAJE SE MARCHO HACIA EL NORTE" se representaba por el propio sujeto abandonando la pantalla en esa dirección.

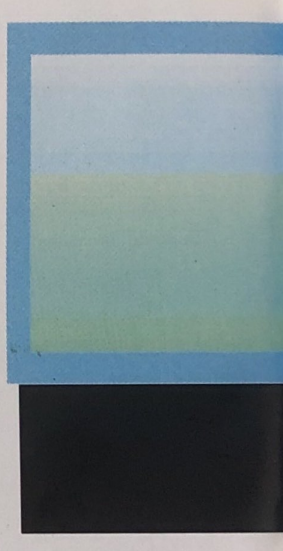
Lucas lo llamó SCUMM, que no hay que confundir con SCUM que significa BASURA. Mediante este genial interface todo se hace mediante el ratón.

En conclusión, merece la pena ir a la tienda y comprar el juego, porque nos va a hacer pasar unos ratos geniales. A la calidad del argumento, comentada antes, se une la facilidad del juego y los gráficos tan cuidados y coloridos que tiene esta aventura gráfica. Otra selección Código Juego para todos vosotros.

# ¡VAMOS DE

## ¿QUÉ TIPO
## DOR NECE
## COMPONER

En la elección de un ordenador personal, además de la actividad musical normal, hay algunas características que hay que valorar con atención. De hecho, no todos los ordenadores del mercado tienen las mismas posibilidades o poseen bibliotecas de software adecuadas para satisfacer todas las exigencias. Si ya posee un ordenador personal, la primera cosa que tiene que hacer es informarse sobre los programas musicales disponibles. Sería realmente inútil emplear grandes sumas de dinero para intentar resolver problemas para los que ya se tiene la solución al alcance de la mano. Si, por el contrario, se ha decidido por una nueva adquisición, o el ordenador personal que usted posee no le satisface plenamente, lo primero que hay que valorar es el nivel de utilización; ¿es aficionado o profesional? Todo esto es de importancia fundamental, porque tanto el ordenador como el software no son nada baratos, por lo que para hacer una elección prudente se requieren algunas consideraciones.



### MEMORIA RAM

La RAM tiene un papel fundamental, ya que de ella depende poder correr determinados programas. Generalmente, los ordenadores personales en los que se instala hardware y software musical de una cierta calidad, tienen memoria suficiente para poder utilizar la mayor parte de las aplicaciones. En cualquier caso se aconseja que tenga como mínimo 1 megabyte de RAM para que nuestras aplicaciones puedan correr con la velocidad que es necesaria en este tipo de aplicaciones y no desesperemos con ellas.

### UNIDADES DE DISCO

Al menos una unidad de disco es absolutamente necesaria para trabajar de manera eficiente. Se aconseja encarecidamente añadir una segunda unidad, cosa que lo haría mas funcional, evitándose fastidiosos y repetidos cambios entre el disco que contiene el programa y el que contiene los datos. Le permitiría acelerar las operaciones de proceso de la información entre el ordenador personal y los instrumentos conectados a él.

### DISCO DURO

Es indispensable que el ordenador tenga disco duro, pues permite contener en él muchos programas, que se usan normalmente permitiendo un manejo de datos más rápido.

### MICROPROCESADOR

Con él logramos una mayor velocidad de proceso de datos, lo que en el ámbito profesional es realmente indispensable.

### UTILIZACION DE INTERFACES MIDI

Como ya se ha dicho anteriormente, la mayor parte de los ordenadores necesitan un interface Midi para su conexión en red. Los únicos ordenadores dotados de serie con este dispositivos son los modelos ATARI y YAMAHA C1.

### TIPO DE ORDENADOR:

Antes de nada os diré, que yo personalmente, me preguntaba por qué todo el

Ordenado

MU

mar

COMPRAS!

DE ORDENA-
SITO PARA
MÚSICA?

# CONEXIÓN MUSICAL

# MIDI

mundo se compra un ordenador, ¿es qué a todos os a dado por jugar a los videojuegos?. Yo, realmente no lo necesite hasta que tuve la necesidad de usarlo en mis composiciones musicales. Recuerdo que de ser un detractor de los ordenadores pasé a ser su fiel exclavo. Sin él, sería imposible que hoy escribiera estas líneas. Pero vamos a centrarnos un poco en el tema y decidamos que ordenador usar.

Si tienes un ordenador, informate de el sofware Midi existente para él si no es así os muestro algunas alternativas según su calidad-precio.

Adquirir un ordenador es una inversión que se debe estudiar muy bien, ya que se deben obtener los programas de aplicación, como un procesador de texto, y además el hardware periférico como la impresora, memoria, disco duro para poder almacenar todos los archivos y documentos elaborados, así como poder usarlos de nuevo sin perder ninguna información creada tras largas horas de trabajo. También, y es el caso que nos atañe, tendremos que conseguir software Midi. Como se puede ver, poco a poco los gastos van aumentando a la hora de adquirir un ordenador, pero aun así, vale la pena la inversión. Por supuesto, hay que justificarla con el uso.

Las personas dedicadas a la música probablemente ya se habrán enterado de la gran cantidad de trabajo que se puede desarrollar mediante un ordenador. Puede escribir e imprimir su música, estudiar partituras, reproducir música creada utilizando un teclado sintetizador etc...

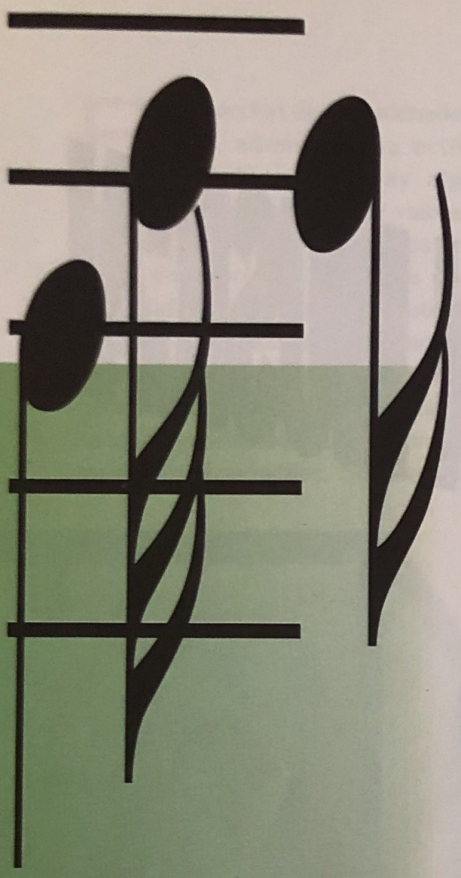
Hoy en día, un compositor puede adquirir un ordenador para la producción de música por medio de un secuenciador, editarla y organizar sonidos de sintetizadores, samplers. Todo ello utilizando la conexión Midi.

Al adquirir un ordenador, por tanto, se deberá investigar cuál es el apropiado. Un punto muy importante atener en cuenta es saber qué cantidad de programas existen en el mercado para el que se pretende adquirir. Entre los ordenadores más populares de aplicación musical, estan: ATARI, MAC, IBM/PC, COMMODORE y YAMAHA.

A fin de elegir el ordenador adecuado para cada necesidad, aparte de estudiar el número y precio de los programas existentes, como mencioné anteriormente, se deberá investigar si éstos son de calidad. . Me refiero a programas secuenciadores,

res con
CHA

ha



**La elección del Ordenador es tarea difícil. Especial cuidado ha de prestarse para que éste se acople a nuestras necesidades.**

que permitan la edición, que posean archivos de sonidos, transcripción de música a papel pautado, etc.

También, se debe tener en cuenta la capacidad de memoria máxima que permite el ordenador y si es barato adquirir memoria extra para él. Cuanta más memoria tiene un ordenador, más secuencias o composiciones se podrán almacenar. Esto es muy útil, especialmente si el músico o compositor tiene la necesidad de tener las obras a la mano para determinados conciertos en directo.

Paso ahora, con más detalle, a describir los ordenadores mencionados anteriormente:

### ATARI

Se le conoce como el ordenador del músico, por la gran experiencia de esta compañía y la dedicación que la misma ha prestado a los temas músicales desde un principio.. Su software Midi es ilimitado y de gran calidad, debido, principalmente a que muchas casas de programación dedican todo su tiempo a buscar y experimentar innovaciones para este ordenador.

Talentos como Madonna, Michael Jackson, Prince, etc... son usuarios asíduos de este ordenador, utilizándolo desde la primera partitura hasta que la pieza musical es estrenada en su último y apoteósico Show. Existen varios modelos de la serie Atari, pero nos vamos a centrar en su último lanzamiento el ATARI FALCON 030. Un revolucionario ordenador en este terreno que aparte de la implementación Midi posee 2 CPUs o microprocesadores. Uno de ellos es el DSP, un procesador de señales digitales, que hace posible la grabación acústica, por ejemplo, la procedente de una guitarra española así como voz humana, ruidos, efectos de sonido, etc, através de la entrada de micrófono.

También, trabaja como el más potentisimo Sampler del mercado y a un precio realmente asequible. Caundo descubrimos, además, que no tenemos que gastarnos más pesetas extra en periféricos, ya que esto lo trae implementado de fábrica, el FALCON 030 se convierte en una buena elección.

Este ordenador es capaz de grabar una fuente acústica a una frecuencia de 50 Khz, más que la calidad incluso que la del disco compacto.

Su sistema operativo es el MultiTos, y es tan fácil de usarlo que tal como te lo compra,s y sin saber informatica, aprietas la carpeta del programa con el ratón y a trabajar. Si deseáis información adicionalm así como precios de este maravilloso ordenador musical, podéis contactar con ASINFO en el número de teléfono 956-813129.

### MACINTOSH

Este tipo de ordenador cuenta con diferentes modelos que tienen como disimilitud la potencia y la velocidad.

Existen muchos programas Midi para este ordenador, aunque sus costos, a diferencia de ATARI, son bastante elevados. También presenta el inconveniente, de tener que comprar periféricos extra para poder operar con el MIDI y el audio.

Pero esto deja de ser una pega, si disponemos de dinero, ya que su rendimiento es bastante óptimo en este campo.

Su entorno de trabajo es a base de ventanas y muy fácil de trabajar con él.

### PCs

Estos ordenadores tienen muchas aplicaciones, como todos sabemos, en cualquier campo que se nos venga a la mente, pero en lo que se refiere a la música, y hasta hace unos años, no existían muchos programas de aplicación musical para ellos. Hoy en día, sin embargo, compañías que se especializan elaborando esta clase de programas para Atari y Macintosh, están traduciendo sus programas para PCs.

Su sistema operativo más difundido, el MS DOS, funciona a través de comandos o instrucciones que se le dan al ordenador para que ejecute la función, pero es tan enrevesado que, posiblemente tendrás que dar clases de este sistema operativo para entenderlo a fondo.

Otra cosa es cuando trabajamos en WINDOWS, que funciona atraves de ventana e iconos, como los sistemas operativos descritos antes en el caso de los Atari y Macintosh.

Los precios de estos ordenadores son moderados y por ellos asequibles a cualquier usuario, existiendo también una enorme variedad de configuraciones posibles.

Una pequeña dificultad es lo aparatosos que son. Para hacer actuaciones en directo, pueden presentar algunos problemas.

## COMMODORE

Aunque la casa Commodore se halla marchado de España, dejando con ello de introducir programas para sus usuarios, todavía se siguen viendo algunos programas que de una forma no ortodoxosa entran en nuestro país. Nos referimos a los programas Midi de la marca Dr. T.

## C1 de YAMAHA

Este ordenador fue diseñado especialmente para aplicaciones musicales. Cuenta ya con el convertidor Midi incluido.

El sistema operativo del C1 es el mismo que el de los PCs. Quiere decir que los programas que funcionan en Pc, tambien funcionan para el C1.

Es un formato portatil; y su pantalla es de cristal liquido (LCD).

Pues bien, esta es la oferta del mercado para todos aquellos que gustan de la música controlada o creada por ordenador. Espero que os haya ayudado en algo conocer estos ordenadores a la hora de adquirir uno. A continuación, vamos a seguir con el estándar MIDI y su formato de trabajo.

## INFORMACION MUSICAL TRANSMITIDA VIA MIDI

### EL CANAL MIDI Y EL MODO

CANAL:

Hábilmente el MIDI transmite una gran variedad de información a un sistema de secuenciación.

El MIDI de cada instrumento incluye 16 canales, através de los que se produce una comunicación o conversación con el ordenador, emviando los diferentes tipos de datos, atraves de estos 16 canales, por separado, por medio de un mismo cable y en el mismo momento.

El concepto de canales Midi es similar al de los canales de Televisión. Muchas estaciones de TV están transmitiendo sus respectivos programas al mismo tiempo y por canales diferentes. Estos canales son recogidos por nuestra antena, así que únicamente tenemos que seleccionar en nuestro receptor de TV, el canal que queremos ver.

De esta misma forma funcionan los canales MIDI.

Un teclado controlador o maestro funciona como una estación de TV y un teclado u ordenador esclavo, hace las veces de receptor de TV. Con la diferencia de que la señal, en vez de ser transmitida por ondas, se realiza mediante cable.

Para un mejor entendimiento aquí va un pequeño ejemplo:

Tenemos 3 sintetizadores conectados de la siguiente manera:

(1) Teclado maestro, encargado de mandar la señal. Función que realiza através de el dispositivo Midi OUT.

(2) Teclado esclavo que recibe la información, unicamente por el canal 1, através del cable MIDIi, el cual está conectado en el dispositivo Midi IN.

(3) Segundo teclado esclavo, que solo recibe por el canal Midi 2.

De esta forma, si el teclado maestro manda la señal através del canal Midi 1; solamente le contestará el esclavo numero 1; y si por el contrario manda la señal por el canal 2, sólo recibirá contestación del esclavo segundo.

En otras palabras, si queremos que funcione el sistema correctamente, tenemos que seleccionar cuidadosamente el canal MIDI.

Con este concepto de los canales Midi, es posible tocar 16 diferentes pistas con un instrumento en cada una, desde un secuenciador y através de un cable Midi.

MODOS:

Como he mencionado anteriormente, es necesario usar el mismo canal Midi si un teclado maestro y un esclavo están tocando conjuntamente, pero hay otra forma de hacer que el esclavo reciba la información. Este modo es llamado OMNI ON y fuerza al esclavo a responder a los 16 canales Midi a la vez, sin importar en qué canal esté transmitiendo el maestro. El esclavo responderá polifónicamente, es decir, que producirá más de una nota al mismo tiempo. Si el esclavo es de ocho voces y el maestro le está transmitiendo diez voces al mismo tiempo, el esclavo sólo responderá a las ocho primeras voces o notas que el maestro le esté enviando. Cuando el esclavo está en el modo (OMNI OFF) solo recibirá información por el canal que anteriormente nosotros hayamos seleccionado. Pero para un mayor entendimiento de la Polifonía y la Monofonía, os diré que son los modos Midi los que deciden cual es la información que será enviada en cada caso. La información Monofónica consta de una sola nota y la Polifónica de más de una nota.

Los instrumentos como pianos y sintetizadores normalmente usan el modo Polifónico.

Mostraremos ahora cuando utilizaremos Poli, Mono, y Omni.

El Midi incluye un selector de Modo, con 4 posibilidades:

MODO 1: OMNI ON, POLI
Recibe la información por todos los canales.

MODO 2: OMNI ON, MONO
Recibe la información por todos los canales pero sólo toca una nota al mismo tiempo.

MODO 3: OMNI OFF, POLI
Recibe sólo por un canal Midi seleccionado en el secuenciador, pero polifónicamente.

MODO 4: OMNI OFF, MONO
Recibe la información por un específico y único canal Midi, pero sólo tocando una nota por canal.

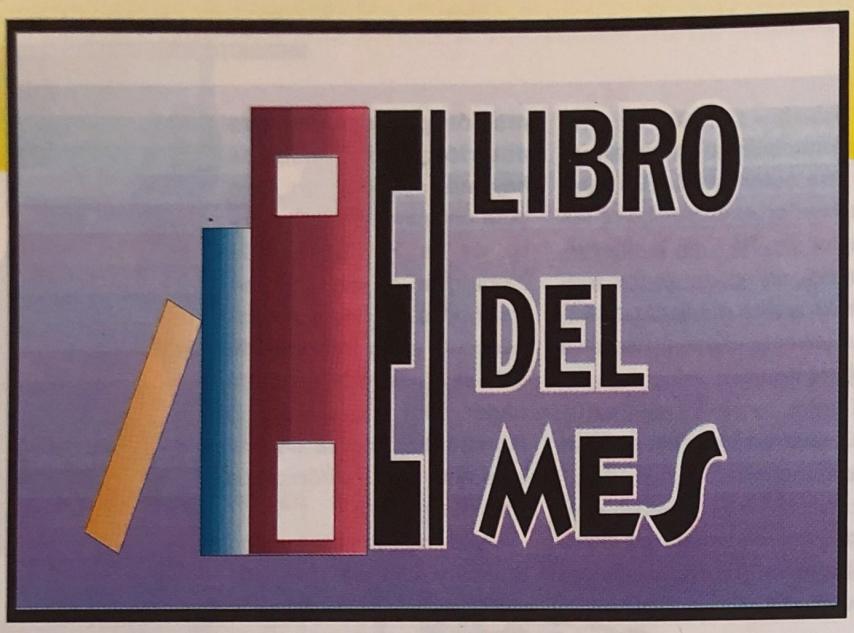
### CONTENIDO DE LA INFORMACION MIDI

El Midi contiene varios bancos de información, que pueden ser usados para transmitir detalles, desde el maestro al esclavo. Como cuando activamos una nota y la liberamos, uso del pedal, etc... De este modo la información se encuentra dividida dentro de los mensajes de canal y mensajes de sistema.

(1) MENSAJE DE CANAL

Este mensaje es enviado, individual-

Continua en página 34






*Como lo prometido es deuda, continuamos el comentario de los manuales de programación que están a tu alcance.*

## FICHA TECNICA

**Título:** Programadores en acción
**Autor:** Susan Lammers
**Editorial:** Anaya/Microsoft Press
**Páginas** 407
**Nivel:** Bajo

## FICHA TECNICA

**Título:** RMP
Revista Microsoft para Programadores
**Autor:** Varios
**Editorial:** Anaya/Microsoft Press
**Páginas** Entre 60 y 100
**Nivel:** Alto

## PROGRAMADORES CON MUCHA ACCION

La vida del programador es dura, pero apasionante.

Esta es la conclusión que se saca tras leer el libro que hoy comento. Una especie de novela por entregas que no enseña el manejo de ningún nuevo paquete de aplicación, ningún lenguaje de última generación, ninguna filosofía reciente sobre programación. Es, simplemente, un compendio de lectura entretenida. Un libro "chicle para el cerebro". Un manual del programador empedernido. Un sitio donde ver tus penas reflejadas. Un muro de lágrimas para el informático.

Y es que, entre sus páginas, un grupo de famosos programadores han contado sus sinsabores, sus penas y sus alegrías.

Muchos estaréis pensando que no merece la pena gastar dinero en un libro de ordenadores, si este no nos enseña por lo menos el BASIC. Nada más lejos de la realidad. Yo pensé lo mismo cuando lo vi en la estantería de la papelería donde lo compré, hace ya cinco años. Empecé a leerlo como un mes después de adquirirlo y porque un día me aburría de solemnidad en casa y el libro "cayó" en mis manos cuando buscaba algo encima de mi armario. No hube leído más de diez páginas cuando ya no me pude desprenderme de su lectura hasta que lo terminé.

Parecía mentira, pero era como si alguien me hubiese estado siguiendo durante mucho tiempo y hubiese escrito un libro con mis experiencias. Allí estaban todos los problemas que yo me había encontrado a la hora de crear un programa, los malos ratos que se pasan cuando un maldito bug se cruza en tu camino y no te deja ni dormir, los esquemas que yo suelo dibujar antes de empezar a meter código en el ordenador, etc. Y todo ello contado por gente que son el no va más de la programación contemporánea, como puede ser el fundador de Microsoft: Bill Gates.

De verdad que os recomiendo su lectura, porque desde ese momento os cambiará, para bien, la forma de trabajar con un ordenador y la filosofía con la que te vas a tomar las cosas a partir de ahora cuando tengas un problema y pienses que eso mismo le pasó a los programadores más famosos de la tierra.

### El libro y sus partes

Como ya he reseñado anteriormente, la obra es una agrupación de entrevistas hechas a distintos personajes que, en mayor o menor medida, han escrito y conducido el curso de la informática moderna.

Entre sus 400 páginas encontramos gente de todo tipo. Tanto los programadores relacionados con el PC como los que están dedicados al MAC, e incluso un programador de vídeojuegos muy, muy famoso, como es Toru Iwatani, creador de PAC MAN, nuestro querido Comecocos.

Tras el bloque de entrevistas, parte principal del libro, viene un apéndice de lo más gracioso. Se trata de una exposición de cosas curiosas. Son pedazos de papel que nos muestran como son las anotaciones que los programadores hacen antes de empezar a desarrollar esa misma idea dentro del ordenador.

Por último un escueto glosario de los términos que se usaron en el libro.

Los programadores

Los personajes entrevistados son los siguientes:

**Charles Simonyi**, de Microsoft, creador, entre otros, del Word y el Excel. **Butler Lampson**, investigador y diseñador de sistemas. **John Warnock**, creador del lenguaje PostScript. **Gary Kildall**, desarrollador del CPM. **Bill Gates**, presidente, fundador de Microsoft y creador del BASIC. **John Page**, creador de la serie PSF. **Wayne Ratliff**, astronauta y creador de DBASE II y III. **Dan Brikin y Bob Frankston** creadores del VisiCalc. **Jonathan Sacs**, que fue el creador del LOTUS 123. **Ray Ozzie** que trabajo para los creadores del VisiCalc y para Jonathan Sacs y su compañía. **Peter Roizen** creador de la hoja de cálculo T/Maker.**Bob Carr**, desarrolló el Framework. **Jef Raskin**, que fue el precursor del sistema Macintosh de Apple computers. **Andy Hertzfeld**, programador de sistemas operativos y creador del que utiliza el Apple III. El siguiente llama la atención por su dedicación expresa a la programación de juegos para ordenador: su nombre es **Toru Iwatani** y fue el programador del famoso Pac Man. **Scott Kim**, diseñador gráfico y programador de imágenes. **Jaron Lanier**, otro programador de juegos, esta vez para el Atari. Y para terminar, **Michael Hawley**, programador y creador de sistemas de audio para la compañía Lucas Films.

Como podéis comprobar se trata de un selecto grupo de grandes y geniales programadores de ordenadores.

De sus extensas y bien documentadas entrevistas, se obtienen innumerables trucos sobre la programación a la vez que aprendemos lecciones de un valor incalculable, que nos ayudarán moralmente en el futuro, cuando nos atasquemos con un programa y nos de por darle una patada al ordenador.

Ya digo, sin ser un libro que enseñe nada de programación, la considero una obra de una riqueza sustanciosa. Os lo recomiendo.

## UNA REVISTA QUE ES UN LIBRO

Esta maravilla editada por la internacionalmente conocida Micrososft Press, nos sirve como guia indiscutible de lo que es la programación correcta, y me gustaría hecer énfasis en esto, de los sistemas software creados por su filial Microsoft Corporation.

Hay mucho escrito, en realidad mares de tinta, sobre esta programación, pero es en esta publicación donde encontramos la referencia más exacta y certera de esos datos técnicos que en muchas ocasiones o no los encontramos o si lo hacemos no resultan fidedignos.

Todo aquel, entre ellos los que usamos el sistema operativo como apoyo en la creación de nuestros juegos, debe poseer una guia de estas características y no tenerla, nos puede conducir a pequeños, pero a la vez grandes, errores en nuestro código.

Eso sí, para moverse con soltura en una obra de estas características, hace falta tener unos conocimientos avanzados del PC. Se aplica esta afirmación en aquellos artículos, la gran mayoría de ellos, en los que se habla de programación en C o en Ensamblador. Sin embargo, hay otros donde se nos ofrece una visión de determinados temas de manera sencilla y exaustiva, de lo que sacamos en conclusión que no sólo se trata de una publicación de referencia sino también de enseñanza.

La obra se edita en dos formatos que son, revista mensual y libro. El primero es el normal de cualquier otra publicación que tiene tirada mensual (aunque comenzó siendo bimestral). La segunda es una recopilación encuadernada de todos los números que aparecieron durante el año anterior.

Aunque algunos se quejan de su precio, hemos de tener en cuenta la naturaleza del contenido. Los artículos, por lo general de extenso tamaño, son de una calidad impresionante, cubriendose todas las áreas de la programación contemporánea.

Las estrellas son, sin duda, la programación y aspectos del sistema operativo MSDOS en sus dos recientes versiones y, por supuesto, WINDOWS, mucho Windows. Así que ya sabéis, cuando os deis una vueltecita por los quioscos echarle un vistazo a RMP, que en muchas ocasiones os va a sacar de ese problema tan quisquilloso.

# La tarjeta VGA (II)

*Es increible la proliferación de tarjetas de vídeo que ha habido durante los últimos años. Al igual que otros muchos componentes del hardware del PC, estas han surgido como setas tras las primeras lluvias. Se ofrecen todo tipo de prestaciones, millones de colores, más lineas en los modos de texto, memoria interna, compresión y descompresión de datos por hardware en tiempo real, etc.*

Una de las cosas más importantes, a la hora de aprender el funcionamiento del adaptador de vídeo, es entender sus posibilidades y restricciones y el porqué de la existencia de éstos.

La más importante de las características del PC, que le ha acompañado durante su toda su historia, es que nunca se ha quedado quieto. Es decir, el hombre no se conformó con aquella lenta máquina de 64K sin disco duro y otros muchos impedimentos, por lo que se dedicó a mejorarlo cada vez que las últimas innovaciones en el campo de la electrónica se lo permitió.

Jamás vimos nuestro ordenador como algo perfecto, como algo que ya no hacía falta mejorar, porque ya cumplía con nuestros requisitos. Incluso hoy en día, en que los PC han alcanzado un nivel de sofisticación elevadísimo, esta carrera por lo perfecto, aunque habría que decir genial, todavía sigue.

Las posibilidades de los ordenadores que adornan la mayoría de las mesas de las empresas y hogares mundiales, superan, con mucho, las exigencias de cualquier usuario medio. Y cabe la pregunta, ¿Para que tanta potencia?.

En muchísimos casos, para nada. Pero en informática ocurre como en todo lo que es vendible y a la vez comprable. El consumismo es malo y ciega a muchos compradores finales.

Es como cuando oigo hablar a un buen amigo mío sobre su afición motociclística con otro compañero. No soportan que nadie supere su motocicleta en cilindrada, aspecto, velocidad, colores, etc. ¿Con los ordenadores?, ocurre más o menos lo mismo.

Si mi 486DX es más lento que tu Pentium, entonces me cambio de ordenador cuanto antes, aunque tenga que empeñar la suegra en el Monte de Piedad y los niños no coman durante un mes. Y después, cuando lo tenemos sobre en casa, montado y funcionando empiezan las preguntas: Bueno, y este ordenador ¿Qué me ofrece de nuevo?.

Con esto queremos decir que nunca viene mal informarse un poco, lo cual ocurre también con los adaptadores de vídeo.

## Mirando atrás

Como decíamos, el PC evoluciona incansablemente, pero de una forma muy especial.

Cuando IBM lanzó el primer PC, la tarjeta de vídeo que incorporaba era la MDA, abreviatura de MONOCHROME DISPLAY ADAPTER, o adaptador de vídeo monócromo (mono=uno, cromo=color). Este apellido no eran en valde, puesto que el monitor que llevaba sólo era capaz de visualizar un color y sólo uno.

La resolución que permitía era únicamente, en modo texto ya que no tenía modos gráficos, de 80x25 (primero se mencionan las columnas y después las filas, no lo olvidéis).

Como veis las posibilidades eran limitadísimas. Pues bien, los super PC´s a los que hacíamos mención antes todavía son capaces de utilizar este modo de vídeo.

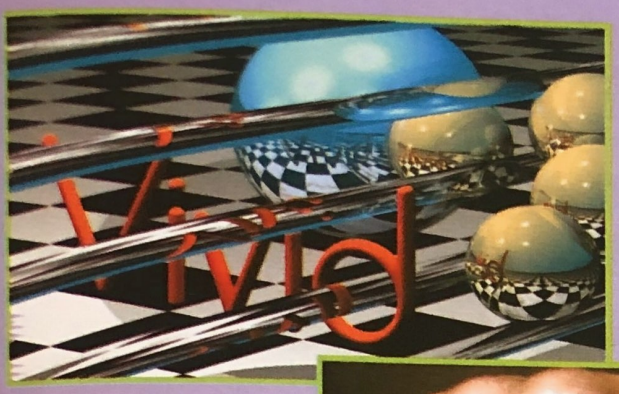
El PC siempre miró hacia atrás cuando se trató de evolucionar. La compatibilidad descendente (tanto en software como en hardware), que así se le llama, ha permitido que todo el software que se creó en años pasados no deje de ser utilizable por las máquinas que incorporan nuevos adelantos.

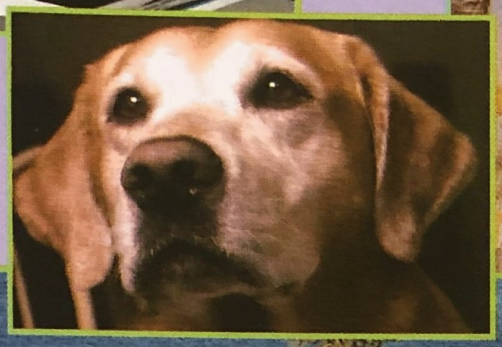
El microprocesador es el principal ejemplo y precursor de la misma, pero también las tarjetas de vídeo creadas por IBM (MDA, CGA, EGA, VGA), han guardado tan educada norma.

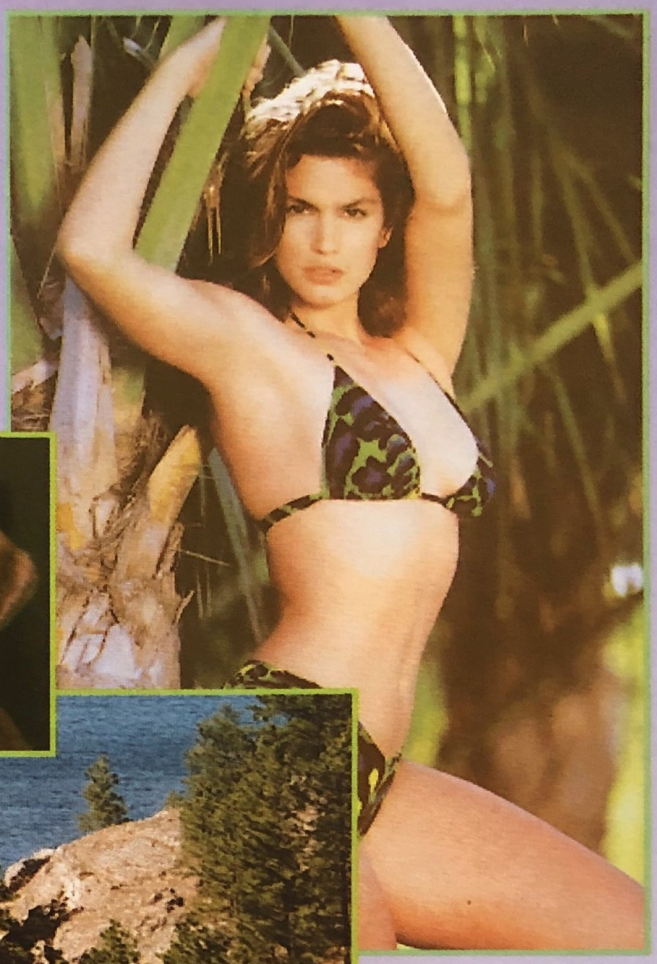
Por un lado este hecho es bueno, pero por otro ha resultado una traba que hemos tenido que arrastrar durante muchos años. Los OEM, sobrenombre que los listillos americanos dan a los fabricantes de hardware, siempre estaban pendientes a lo que IBM lanzaba, para, rápidamente copiarlo en la medida de lo posible y abaratar el precio.

Al no ser tarjetas-copias exactas de lo que IBM había inventado, porque el Copyright lo impedía, los adaptadores de vídeo de terceros fabricantes no podían nunca llegar a ser 100 por 100 compatibles. Siempre existía alguna incompatibilidad que las hacía diferentes de la norma IBM.

Lo que ocurre es que, por lo menos, en aquel entonces alguien marcaba





**LA TARJETA VGA DOTÓ AL PC PARA LA FOTOGRAFÍA, CULTURA, EDUCACIÓN, INFOGRAFÍA, INVESTIGACIÓN, ETC...**

el compás, alguien golpeaba el tam-tam de la compatibilidad, alguien decía lo que los demás copiaban. Pero ahora, los OEM se han disparador cada uno por su lado con la llegada de la SuperVGA.

Al margen de esta nota, sigamos con lo nuestro. Las tarjetas de vídeo que salían nuevas al mercado por parte de IBM siempre tenían dentro la circuitería necesaria para emular los modos de vídeo de las tarjetas que habían sido creadas con anterioridad. De esta forma, se aseguraba que el software siempre sería compatible con los nuevos productos hardware. Así, las tarjetas CGA podían hacer lo mismo que la MDA, simplemente conmutando por software, es decir desde programa, al modo de vídeo que pertenecía a la tarjeta que se pretendía imitar. Se lograba de esta manera que lo antiguo no callera en desuso. Así de fácil.

## Los modos de vídeo hardware y el BIOS.

Pasemos a otro punto de vital importancia.

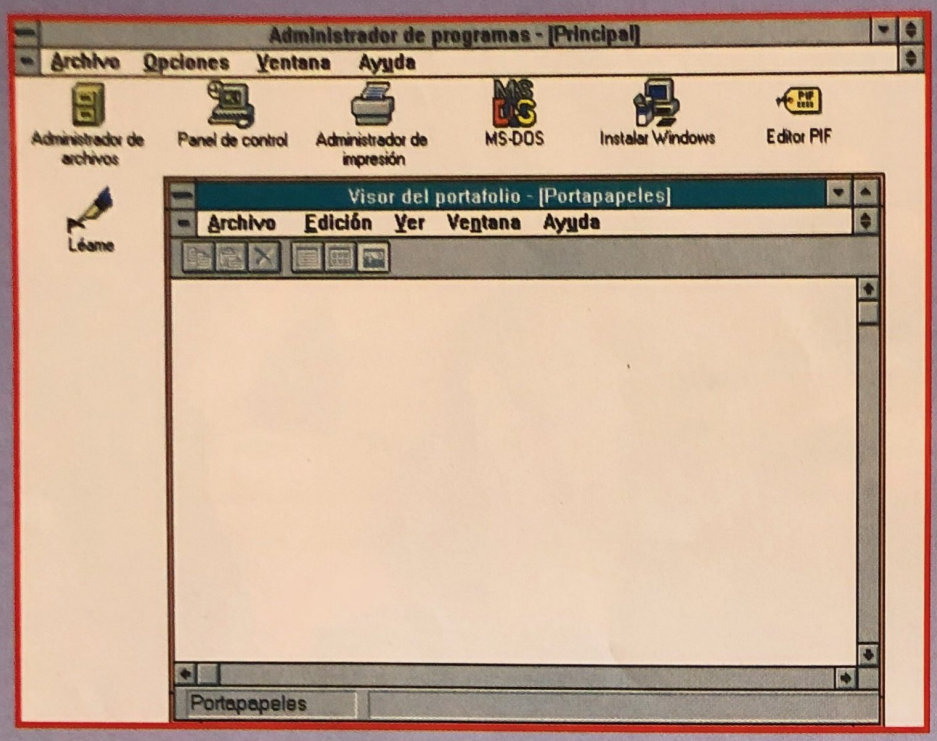
Como hemos explicado, los OEM que "copian" el sistema IBM, en realidad no pueden calcar exactamente la circuitería de la empresa inventora por lo que tienen que hacerlo que funcione igual, pero de forma diferente.

Toda tarjeta de vídeo, por ejemplo la VGA, permite, mediante la conmutación de modos gráficos, acceder a los diferentes "estilos" de presentación visual de la información en pantalla, de acuerdo con las necesidades de nuestros programas. Los modos se se pueden agrupar o dividir de varias maneras, pero la principal sería:

### Modos alfanuméricos o de texto

La información en pantalla son siempre caracteres (números, letras, símbolos).

Los únicos gráficos que pueden mostrarse en estos modos de vídeo son aquellos que se logran a base de la construcción de los mismos mediante los pocos símbolos o




**Los entornos gráficos como Windows, han sabido aprovechar las posibilidades de la tarjeta VGA.**

Pseudográficos que están permitidos.

La resolución de estos modos se mide por filas y columnas de caracteres y nunca por pixels. Además, cada modo de vídeo de texto permite una configuración de tamaño de celdilla de carácter diferente. Así, hay caracteres con pocos puntos de definición (8x8 pixels por carácter) y los hay de más calidad (9x14 pixels de definición). Ver figura.

Modos gráficos: El PC se libera en estos modos, de la cautividad de los modos de texto, ya que aquí si se se permiten complicadas imágenes de cualquier tipo. La resolución se mide en pixels por filas y columnas ya que es la unidad mínima de información de una imagen. De esta forma, tenemos las resoluciones de las que tanto habéis oído hablar en todos sitios; 640x480, 1024x768, 320x200 etc.

La cicuitería de vídeo permite la conmutación entre dichos modos mediante la programación a nivel de hardware de la tarjeta, cosa que podemos realizar siempre que queramos. Los modos de vídeo estándar se logran mediante la introducción de unos parámetros prefijados en los distintos registros del adaptador. Con ello se logra el ajuste de todos los componentes de forma que estos miran la información que esta dentro de la VRAM (memoria de vídeo) de diferente forma en cada caso. Esto lo veremos más adelante, de todas formas, os adelantamos que si utilizamos siempre la misma memoria, los bytes que introduzcamos en ella, en ocasiones serán tomados como descriptores de la fisonomía de un píxel (color del mismo) y en otras como el código ASCII de un carácter alfanumérico, según en el modo de vídeo en que nos encontremos en ese momento. Esto es harina de otro costal y ya lo estudiaremos con más profundidad. Tan sólo debéis saber que el hardware de la VGA contiene unos componentes llamados REGISTROS que se pueden programar para que actúen de diferente manera. Gracias a esta programación se pueden hacer maravillas con la tarjeta, como el scroll que es tan necesario en nuestros juegos.

El problema está en que, como explique más arriba el hardware de la tarjeta puede nos ser todo lo compatible con las especificaciones de IBM como sería desear. Por tanto, algunas acciones con él, pueden llevarnos a la catástrofe cuando lo transportamos a otro ordenador. Esta precaución siempre habrá que tenerla en cuenta.

Una forma de evitar estas trabas es utilizar el BIOS de la tarjeta para que él, que ha sido escrito por el propio fabricante de la tarjeta, se ocupe correctamente de este proceso de cambio de un modo de vídeo a otro.

Por otro lado, para quitaros un poco el miedo, os diré que gracias a Dios, las tarjetas son lo suficientemente compatibles como para que la mayoría de los programas que hagamos a nivel de hardware con ellas, funcionen sin problemas. En resumen, hay dos formas de acceder a la tarjeta de vídeo: mediante el BIOS con llamadas a las interrupciones que están documentadas y que veremos aquí una a una y mediante la programación directa de los registros de la VGA mediante el lenguaje ensamblador o C y que como he dicho anteriormente, puede, en algunos casos, darnos problemas con las pequeñas incompatibilidades.

Los modos estándar

En esta serie de artículos vamos a estudiar tan sólo un modo de vídeo. El que se conoce como 13h (a aquellos que no conozcan la numeración hexadecimal les aconsejamos que se lean el curso de ensamblador). Este modo gráfico, exclusivo de las tarjetas VGA y MCGA (nota: exclusivo significa que se implementaron en ellas como novedad), que permite una resolución de 320x200 pixels y 256 colores de entre una paleta posible de 262.144 colores, es el más utilizado en la actualidad por los diseñadores de vídeojuegos. ¿Por qué?, sinceramente por que no hay otro. Los modos SuperVGA presentan dos problemas principales: La incompatibilidad, que hace que sean necesarios los drivers de la tarjeta en cuestión y la lentitud de proceso. Lo primero se puede salvar, aunque hace que cualquier juego se vea engordado innecesariamente con los drivers de todas la tarjetas SVGA existentes. Lo segundo es un problema muy difícil de evitar y en muchos juegos esta lentitud hace que no se puedan jugar con una mediana calidad de animación. Por ello, y en espera de que las tarjetas VESA se hagan dueñas del mercado y más rápidas, los programadores no podemos más que apretar los dientes y conformarnos con lo que nos ofre-

ce el PC por ahora.

Además, la programación del modo de vídeo 13h nos ofrece grandes alicientes que ya veréis como no os aburren.

Sin embargo, para conocer el sistema de vídeo del PC es necesario que estudiemos un poco el resto de los estándares gráficos que la VGA es capaz de emular. Ver figura.

Los modos de vídeo, como ya sabemos, se dividen en dos. Los alfanuméricos, que a partir de ahora los vamos a llamar simplemente "de texto" y los gráficos. En la tabla también están listados los colores que cada modo de vídeo es capaz de sacar en la pantalla. Si leísteis el artículo anterior (Código Juego 1, Junio 1994) ya sabéis que la cantidad de colores y la resolución están relacionadas directamente con la cantidad de memoria de que debemos disponer en la tarjeta de vídeo; VRAM.

A estos modos de vídeo se les llama también modos del BIOS, porqué son los que se logran mediantes simples llamadas a éste desde nuestros programas. El número hexadecimal que aparece a la izquierda de estos son los que utilizaremos cuando llamemos al servicio de interrupciones de sistema ROM de la tarjeta.

La VGA, sobre las demás tarjetas existentes con anterioridad, aporta el modo de vídeo motivo de nuestros artículos (13h) y los de 640x480 en 2 y 16 colores (11h y 12h), como novedad sobre los que y aportaron otros como MDA, EGA Y CGA (modos del 00h al 10h).

## Los "otros" modos

En la figura 3 se puede ver una tabla con las características de los diferentes modos de vídeo permitidos por la tarjeta VGA. Insisto en que éstos son los modos estándar o del BIOS. Existen otros modos que han sido creados por diferentes programadores para satisfacer las necesidades especiales de la aplicación que en ese momento estén realizando. Los creadores de estos modos, por lo general programadores de vídeojuegos, reprograman el hardware para lograr que los modos estándar sean más potentes.

Por lo general, cuando arrancamos el PC este entra en modo 03h (Texto, 80x25, 16 colores) que es el más apropiado para trabajar con el MSDOS y sus comandos. Windows, sin embargo, necesita más capacidades gráficas y por ello usa todos los modos que la tarjeta de vídeo instalada pueda permitir, incluso el Truecolor (16 millones de colores). El modo de entrada de Windows, antes de que lo cambiemos por otro, es de 640x480 en 16 colores.

Las tarjetas VGA no hubiesen evolucionado tanto, si no llega a ser porque los sistemas operativos contemporáneos (OS2, Windows, Windows NT, Chicago) están necesitando, cada vez más, lo que se conoce como GUI o Entornos Gráficos del Usuario. Se intenta ponérselo fácil -ya era hora- al usuario mediante iconos y el uso exaustivo de gráficos y el ratón. Se rompe, de esta forma, con el MSDOS y sus lineas de comandos, acercando más el ordenador a los usuarios que no tienen muchos conocimientos de informática.

Los modos BIOS, deben ser utilizados siempre que se pueda. En primer lugar, nos ahorramos el trabajo de tener que reprogramar los registros y en segundo lugar nos evitamos las posibles incompatibilidades que esto supone. Por esto, siempre que veamos que los modos de vídeo preestablecidos son justo lo que necesitamos utilizaremos estos mismos.

## Los modos y los juegos

Por lo general, el programador de vídeojuegos buscará la máxima profesionalidad en la presentación visual. Esta norma la seguiremos siempre, porque hoy en día, los vídeojuegos no son catalogados como aceptables si no cumplen unas mínimas reglas gráficas. Esto, nos obliga a luchar con los modos gráficos (VGA y SVGA principalmente) hasta sacar de ellos todo el jugo. No en vano, muchos programadores de grandes compañías diseñadoras de juegos están dedicados exclusivamente a la creación visual de los mismos, desarrollando el mejor interface con las tarjetas de vídeo que sea posible.

El único freno es que no se pueden utilizar lo que yo denomino "modos prohibidos", es decir, los SuperVGA. Desgraciadamente, hasta que la estandarización no llegue y la velocidad de proceso de estas tarjetas se mejore los programadores tenemos las manos atadas y no podemos correr el riesgo de que nuestro juego, tras un largo proceso de diseño y creación, no funcione en todos los ordenadores. De todos modos, cuando la programación de los modos estándar VGA, haya sido cubierta en estas páginas, comenzaremos una segunda entrega con la programación de los modos SVGA para que aquellos valientes, que se atrevan a adentrarse en ellos, encuentren ayuda en Código Juego.

Por ello, el mejor de los modos de vídeo que se pueden utilizar y de hecho es el que se utiliza en un 95% de los juegos es el 13h y a él, y todas sus posibilidades, dedicamos esta serie de artículos. Además, intentar explicar todos los modos existentes nos retrasaría enormemente, teniendo en cuenta que los modos gráficos CGA y EGA están ya en desuso por las mejores prestaciones que la VGA y no ofrecen nada interesante al software lúdico. Sin embargo, si se harán reseñas a los modos de texto, porque sí son útiles en los juegos a la hora de presentar información alfanumérica en pantalla.

En el próximo número empezaremos a ver como usar el BIOS de vídeo incorporado en la tarjeta del adaptador. Recuerda que cuando comencemos a programar utilizaremos el lenguaje ensamblador, así que no olvides seguir el curso que también esta en nuestras páginas. ¡Hasta pronto!


**RECUERDA QUE EN**

**PUEDES PONER TU ANUNCIO POR PALABRAS O TU OFERTA DE TRABAJO**

**TELEFONO/FAX 956-813129**

mente por los diferentes canales Midi; para especificar el instrumento en el sistema y por lo tanto su correspondiente efecto que recibe el canal. Los mensajes de canal incluyen: nota activada y desactivada, modulador de sonido (pitch bend), pedal activado o desactivado, etc...

Estos mensajes de canales se dividen en dos, MENSAJE DE VOZ y MENSAJE DE MODO.

(A) MENSAJE DE VOZ

* INFORMACION DE LA NOTA

La información de la nota nos indica, cual es la tecla que se esta presionando, y su correspondiente liberación.

* CAMBIO DE PROGRAMA

El cambio de programa lo usamos, para indicar al teclado esclavo que vamos a cambiar un sonido. Los sintetizadores, organos, pianos electrónicos y samplers, tienen en su memoria muchos sonidos. Con este cambio de programa los músicos pueden decidir el sonido que van a usar en cada momento.

* CAMBIO DE CONTROL

Un cambio de control puede añadir, sutilmente, una atenuación, realizando cosas como; modulación (vibración y temblor), pedal de sostenido, pedal de tono y transportador. Este mensaje no es usado en todos los instrumentos Midi. Un piano electrónico, el cual manda y recibe la señal del pedal de tono, normalmente no necesita el transportador. Pero si el teclado maestro tiene transportador (portamento) el piano no responderá a la información. Para saber cuales son los controles específicos de cada instrumento, hay que ojear la guia de implementación Midi, que trae cada uno. Más adelante os enseñaré a leer una tabla de implementación Midi.

* PRESION DEL TECLADO (AFTER TOUCH)

Los sintetizadores y samplers pueden controlar el vibrador, el brillo, el volumen, etc... simplemente presionando una tecla. La presión del teclado puede usarse para transmitir algunos efectos via Midi. Por supuesto, el instrumento esclavo, recibirá y contestará inmediatamente a la información recibida.

* MODULADOR DE TONO (PITCH BEND)

Si el teclado maestro tiene PITCH BEND, la información puede ser mandada via Midi. Una vez de nuevo, el esclavo recibirá y contestará inmediatamente a la información recibida.

(B) MENSAJES DE MODO

Como os mencione antes, el Midi tiene 4 MODOS, y estos son usados para cambiar los modos del esclavo. Algunos sintetizadores o pianos electrónicos están en el MODO 1 (OMNI ON, POLI), cuando los encendemos, pero deberás ponerlo en el MODO 3 (OMNI OFF, POLI) para usarlo con un secuenciador. Te preguntaras, ¿esto por qué?, por que algunos secuenciadores mandarán automáticamente, al esclavo, un mensaje de MODO 3. El maestro hará que el esclavo cambie sus parametros, en un abrir y cerrar de ojos.

(2) MENSAJE DE SISTEMA

El mensaje del sistema, se enviará, através de todos los teclados conectados al mismo cable Midi.

Por ejemplo, se usa para sincronizar un secuenciador y una caja de ritmos, para hacer que empiecen a tocar, sincronizadamente los dos instrumentos a la vez y también que su respectiva parada (stop) sea idéntica. Se usan, además par indicar un problema del sistema via Midi.

Estos mensajes se llaman SISTEMA EXCLUSIVOS (System Exclusive), ya que son mensajes exclusivos de cada fabricante de instrumentos Midi. Cada uno tiene un numero de identificación, con el cual se reconoce la marca del teclado que envía la señal y sus características. Con el mensaje exclusivo, es posible transmitir sonidos entre sintetizadores, o cambiar sus parámetros.

Los mensajes de sistema son aplicados a un sistema Midi en conjunto, no necesitan estar asignados a un canal de Midi específico. Estos mensajes sirven para manejar a control remoto los instrumentos Midi con un controlador maestro. Estos mensajes son muy útiles en actuaciones en vivo. Los mensajes de sistema se dividen en tres y son:

(1) MENSAJES DE SISTEMA COMUN

(2) MENSAJES DE SISTEMA DE TIEMPO REAL

(3) MENSAJES DE SISTEMA EXCLUSIVO

Veámoslos más a fondo.

(1) MENSAJES DE SISTEMA COMUN

Los mensajes de sistema común son , generalmente, utilizados para controlar secuenciadores, ordenadores y cajas de ritmos. Los mensajes de sistema común son:

(A) MARCADOR DE POSICION DE LA SECUENCIA

Este mensaje es usado en conjunto con los mensajes de sistema de Tiempo Real. Cuando se desea empezar a tocar una composición, a un determinado compás en un secuenciador, con sólo marcar el número de compás en el secuenciador, sea este físico o en programa por medio de un ordenador, la composición empezará a tocar desde ese compás. Los aparatos con Midi interconectados al secuenciador, como caja de ritmos por ejemplo, empezarán a tocar desde ese punto. Esto fue implementado para no tener que iniciar desde el principio de la composición cada vez que se desee escuchar una sóla parte de ésta.

(B) SELECCION DE CANCION

Este mensaje es parecido al mensaje de cambio de programa, con la diferencia de que al mensaje de selección de canción no necesita asignársele ningún canal Midi. La selección de canción es muy útil en concierto donde en lugar de cambiar a la siguiente canción instrumento por instrumento, con sólo oprimir un botón en el teclado maestro todos los instrumentos se cambiarán a la canción correcta.

Llegamos de esta forma al final del espacio que tenemos asignado dentro de la revista.

En el siguiente número, continuaremos donde lo hemos dejado este mes. No olvidéis comprar la revista siguiente porque si no lo hacéis os voy a perseguir incansablemente.

Pronto habrá música en los juegos.

# EN CÓDIGO JUEGO

# Buscamos programadores y colaboradores

¿Sabes hacer algo con tu PC, Atari o Amiga y quieres compartirlo con los demás?

¿Quieres ganar dinero haciéndolo?

Si te animas verás tu artículo publicado en nuestra revista.

Envíanos tu artículo, tu disquete, tu truco, tu listado por pequeño que sea, incluso tu juego completo, y si es seleccionado por la redacción de la revista, pasarás a ser uno de nuestros colaboradores, echándote dinerito en el bolsillo. Contacta con nosotros en el número de teléfono que indicamos abajo o por correo.

CODIGO JUEGO
Ajatel S.L.
Apdo. 114
11520 Rota (Cádiz)

Teléfono 956-840246

# Lenguaje C

## Ficheros, Funciones, Bloques, Variables,... Formato de ficheros y bloques en C

José A. Brihuega Parodi

2a Parte

**Continuamos este apasionante curso sobre el Lenguaje más versátil que existe. En esta ocasión empezamos con el estudio de las sentencias y de su sintaxis. Coged lápiz y papel. No perdáis comba.**

Es fundamental para comenzar a programar, conocer perfectamente la forma en que se estructuran los programas en el lenguaje elegido.

La estructura de los programas en C mantienen la sencillez en la forma que caracteriza a este lenguaje: todo está orientado a lo práctico, y todo tiene una correspondencia directa con lo que posteriormente obtendremos en el código objeto. Es por esto por lo que se suele decir que C es más cercano al ensamblador que otros lenguajes.

### Funciones

La estructura básica en C es la función. Se puede decir que un programa en C es básicamente una serie de funciones, de las cuales *una y sólo una de ellas* tiene el nombre de *main()*, y hace el papel de programa principal. A partir de ahora, la presencia de los paréntesis diferenciará a las funciones del resto de objetos del lenguaje.

Las funciones que componen un programa se agrupan en uno o varios *ficheros*, cada uno de los cuales, tras ser compilado, se convertirá en un módulo de código objeto. Estos módulos, tras el proceso de enlazado *(link)*, conformarán el programa ejecutable. Por tanto, la separación de las funciones en diferentes ficheros tiene influencia directa en la ejecución posterior del programa.

### Independencia Funcional

La gran ventaja que obtenemos de ello es la total independencia entre los objetos contenidos en cada fichero. Cada fichero se compilará por separado, por lo que no habrá que preocuparse de que uno interfiera en el trabajo de los otros. De hecho, los objetos que pertenezcan a uno serán totalmente invisibles para el resto, excepto aquellos que hayamos declarado explícitamente como externos.

De hecho, nada impide que, por ejemplo, cada fichero sea desarrollado por un programador diferente: cuando un programador tuviese que utilizar una función externa de otro, ni siquiera tendría que conocer cómo trabaja internamente dicha función. Esto introduce el importante concepto de la función como caja negra al que dedicaremos un capítulo.

## Bloques

Una función está formada por:

a) *La declaración de la función*, que contiene el tipo, el nombre de la función y la lista de parámetros.
b) *Un bloque*, que es el cuerpo de la función.

El concepto de bloque es otro de los fundamentales. Todas y cada una de las instrucciones del programa estarán incluidas en bloques, que serán tratados como unidades independientes. En cualquier parte donde haya una instrucción, éstase podrá sustituir por un bloque de instrucciones "encapsuladas". Así, los bloques se irán anidando en niveles independientes: las variables pertenecientes a un bloque *(locales)* serán invisibles fuera de él, pero no para los bloques que este contenga.

Un bloque está formado por:
*Una llave* ({) de apertura, que señala el comienzo del bloque.
*La zona de declaración de variables*, donde se definen las variables locales del bloque.
*La zona de instrucciones*, que contendrá las acciones a ejecutar, así como otros bloques anidados.
*Una llave (}) de cierre*, marcando el final del bloque.

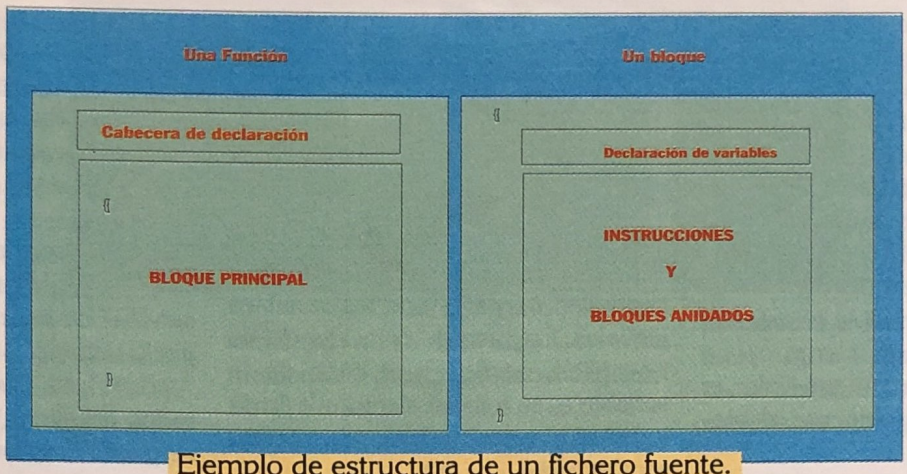
Anteriormente hemos señalado que el cuerpo de una función es en sí un bloque, es decir, la sucesión de bloques anidados tiene su término en el "superbloque" que forma la función en sí. De esta forma, hay una correspondencia casi directa entre los conceptos de bloque y de función.

## Variables globales y locales

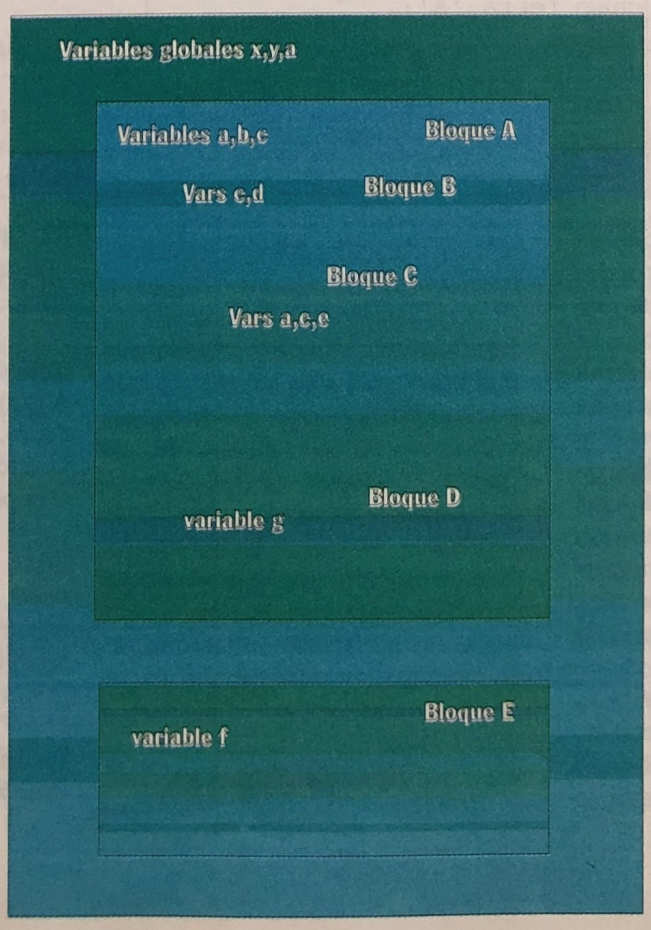
Una variable es una porción de memoria donde guardaremos un dato a utilizar en el programa. Cada variable tendrá un nombre que la identificará dentro de su bloque, así como un tipo que identificará los valores que puede aceptar (p.e., si aceptará caracteres o números enteros).

Como se ha dicho anteriormente, cada variable local (i.e., definida dentro de un bloque) sólo será accesible dentro de él. Esta importante premisa, que garantiza la independencia, supone un inconveniente cuando necesitamos que determinados datos sean accesibles en todos los bloques.

Es por ello que, al principio de cada fichero, se habilita una zona donde se declaran las variables globales, que serán accesibles en cualquier punto del fichero.



Ejemplo de estructura de un fichero fuente.



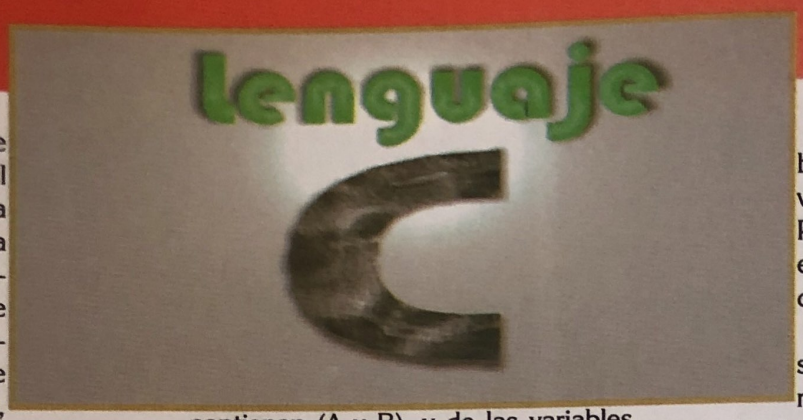

Es muy importante que sólo se haga global una variable cuando esta característica sea totalmente imprescindible. Una gran cantidad de variables de este tipo falsearía la independencia de los bloques, haciendo nuestro programa más propenso a errores y conflictos entre variables.

Que una variable sea global en un fichero, no la hace accesible en el resto de los que forman el programa, excepto en aquellos en los que haya sido declarada como externa. Sin embargo, es éste un tema más complejo que se tratará cuando se expliquen, entre otros, los archivos de cabecera.

## Más sobre variables

Hay una gran diferencia entre declarar una variable y reservar memoria para ella. Una particularidad de C es que deja total libertad al programador para decidir el lugar y el momento en que se reservará y se liberará el espacio para una variable.

*Reservar memoria* es abrir una zona de memoria en la cual se almacenará el valor de la variable. Al realizar dicha operación, los datos que contenía dicha zona de memoria no se eliminan, por lo tanto, no se puede prever el valor inicial de una variable recién creada. Olvidar este detalle suele generar errores imprevisibles y de difícil localización.
*Liberar memoria* es dejar el espacio ocupado por dicha variable a disposición de cualquier otra necesidad de memoria. Los datos que contenía la variable no se eliminan, pero serán sobrescritos cuando sea necesario.




En el caso de una variable local, el espacio de memoria necesario para dicha variable se reserva cuando empieza a ejecutarse el bloque al que pertenece, y se libera al salir de dicho bloque. Por tanto, la variable no es accesible ni antes ni después del bloque; tampoco se podrá acceder a su antiguo valor la siguiente vez que se ejecute el bloque.

Al contrario, una variable global se reserva al comenzar el programa y se libera al terminar. Por ello, los valores que asignemos se mantendrán durante toda la ejecución, aunque sólo será accesible en el fichero que la definió, sin perjuicio de aquellos donde se referenció como externa.

## Conflictos entre variables

Una de las ventajas de la independencia entre bloques es la posibilidad de utilizar nombres iguales para variables distintas en distintos bloques. De esta forma, eliminamos la preocupación que existe en otros lenguajes en el sentido de evitar los conflictos en la denominación de las variables.

Cada variable que se defina con igual nombre en bloques diferentes será a todos los efectos un objeto distinto, con espacio de memoria aparte y valor propio.

Cuando se haga referencia a un nombre de variable, se tomará la que existe dentro del mismo bloque o, en su defecto, en un bloque mayor que lo contenga en un orden del más interior al más exterior. Si no existe en ninguno de los bloques anidados, se toma la variable global con dicho nombre en el propio fichero. En el caso de que no exista una variable global con dicho nombre, se producirá un error al compilar, indicando que se hizo referencia a un nombre de variable no definido.

En el esquema anterior, p.e., desde el bloque C serán accesibles, además de sus propias variables locales, las de los bloques que lo contienen (A y B), y de las variables globales. En caso de coincidencia de nombres, como en el caso de la variable a, se tomará siempre la "más local" al bloque desde donde intentamos acceder.

Obsérvese, por ejemplo, la situación de la variable global a, la cual se encuentra "tapada" por la del mismo nombre en el bloque A, lo que impedirá su acceso en dicho bloque y en los que de él dependen (B, C y D). Sin embargo, no tendrá problema en el bloque E.

Téngase en cuenta también que desde el bloque E sólo se podrá acceder a la local f y las globales x,y,a, pues el resto de los bloques están totalmente aparte.

Aunque, a primera vista, esta forma de trabajar pueda parecer un poco complicada, lo cierto es que, sin esta característica sería casi imposible programar algo medianamente complicado. Esta independencia entre los bloques conlleva, principalmente, dos ventajas fundamentales:

-- Por muy complicado y extenso que sea el programa, sólo tendremos que preocuparnos en cada momento de unos pocos bloques y unas pocas variables. Podremos concentrar nuestros esfuerzos en unas pocas líneas de código.

-- El espacio en memoria para el contenido de cada variable sólo está reservado dentro de su bloque, lo que permite gestionar mucho mejor la memoria disponible.

```
unsigned x;
int x,y,z3,valor;
char letra='A';
unsigned long valor_largo=1000000;
double var_doble_precision;
float real_1, real_2, real_3;
```

**Ejemplos de declaración de variables**

## Tipos Básicos

El tipo de una variable define el conjunto de valores que puede aceptar. Por tanto, es fundamental estudiarlos en profundidad.

Los tipos básicos son los que están predefinidos por el lenguaje, y servirán de base a los tipos definidos por el usuario.

Los tipos base en C son todos escalares, es decir, cada valor se puede representar por un número entero o fraccionario. Así, por ejemplo, un carácter se representa internamente por su correspondiente código ASCII, que es un número entero.

Hay cuatro tipos fundamentales, identificados por la palabra clave que los define en el lenguaje:

int: números enteros.
char: caracteres.
float: números reales, representados en notación de coma flotante.
double: Números en coma flotante, con doble número de dígitos significativos.

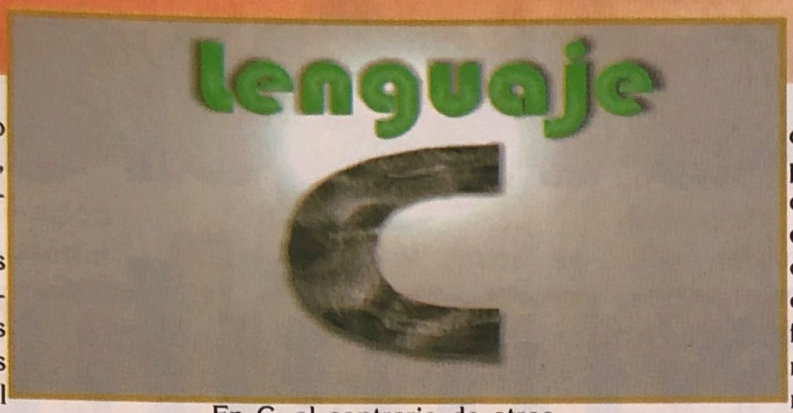
Existe también una serie de modificadores que alteran el rango de valores aceptados por los tipos base:

short int, o simplemente short: números enteros con menor rango de valores.
long int, o también long: números enteros con mayor rango de valores.
unsigned int, o simplemente unsigned: números enteros sin signo.
unsigned long: enteros grandes y sin signo.

El rango y el espacio de memoria ocupada por cada tipo varía según el compilador y el sistema operativo, es, por tanto, un tema a tratar cuando se comparen los diferentes compiladores existentes.

Por la especial forma que tiene C de tratar internamente los tipos, existen algunos modificadores no tan sencillos (como, p.e. unsigned char) que son materia para un capítulo más avanzado.

En el cuadro se pueden ver varios ejemplos de declaración de variables.




Obsérvese que, como todos los comandos en C, cada declaración se debe terminar con un punto y coma.

También hay ejemplos de declaración de varias variables del mismo tipo, separadas por comas, así como variables que toman un valor inicial al ser definidas.

Los nombres de variables pueden contener caracteres alfabéticos y numéricos, además del signo de subrayado (_). El primer carácter del nombre np puede ser un número. La longitud máxima que puede tener el nombre dependerá del compilador con el que trabajamos, aunque habitualmente suele ser de 32 caracteres.

## Comentarios

Las notas y comentarios son un elemento importante para cualquier lenguaje de programación. En lenguaje C, por su complejidad, cobran aún más importancia. Aunque, en principio, suele resultar un tanto engorroso comentar los programas, el esfuerzo será recompensado a la hora de modificarlos y depurarlos, así como cuando se desee que otras personas lean nuestros trabajos.

```
/* Esto es un comentario */
/* Los comentarios pueden ocupar
 varias líneas */
/* Los comentarios anidados
/* se permitirán o no */
 dependiendo del compilador */
```

El campo comentario

En C, los comentarios se ponen entre los caracteres /* y */. Todo lo que se ponga entre ellos será ignorado por el compilador. En el cuadro adjunto se presentan varios ejemplos de comentarios.

## Tipos y parámetros de las funciones.

Las funciones, al igual que las variables, también se identifican por su tipo. El tipo de la función es el valor que dicha función devolverá al ser invocada. El valor devuelto será el que se indique dentro de la función con la sentencia return.

En C, al contrario de otros lenguajes, no se hace distinción entre funciones y procedimientos. Siguiendo el enfoque práctico que tiene el lenguaje, todos los procedimientos son funciones. El valor devuelto se usará o no según lo necesitemos.

Por otro lado, los parámetros son los valores que se le pasarán a la función al ser llamada. Cada parámetro funciona dentro de la función como si de una variable local se tratara.

Es importante entender que en C todos los parámetros se pasan por valor, es decir, a la función se le suministra una copia aparte del dato, de tal manera que cualquier cambio que la función se haga sobre dicho dato no tendrá reflejo fuera de ella. A veces, nos es necesario que no sea así. Cuando se estudien los punteros, se verá la forma de conseguirlo.

El tipo y los parámetros de la función se definen en la cabecera de la función: primero el tipo, luego el nombre de la función y, por último, los parámetros separados por comas, y entre paréntesis. Cada parámetro irá precedido de su tipo. En el cuadro siguiente se presentan varios ejemplos de cabeceras de funciones.

```
int convertir(char letra_1, char letra_2)
double raiz_cuadrada(double operando)
void escribir(char letra)
unsigned long espacio_libre(void)
```

Ejemplos de funciones válidas

Como se observará, las cabeceras, al contrario de las instrucciones, no deben terminarse con un punto y coma.

Un punto interesante es la aparición del tipo void (vacío) para representar la ausencia de parámetros, o la ausencia del valor devuelto. Ha de tenerse en cuenta que la presencia de este tipo especial es puramente informativa, pues, internamente, el compilador tomará el tipo int para las funciones void, y dejará los paréntesis vacíos () cuando se use dicho tipo en los parámetros. La utilidad de void reside en que posibilita al compilador para controlar si estas funciones son llamadas correctamente durante el programa,evitándose errores; por tanto, debe usarse. Además, su utilización hará mas comprensibles los programas para quien los leyese.

## Llamadas a funciones

Para ejecutar una función desde otra o desde sí misma (recursividad), se usa el nombre seguido, entre paréntesis, por los valores a asignar a los parámetros. Son los paréntesis los que la identifican como función, y no se deben omitir. Si no existen parámetros, se dejan los paréntesis vacíos.

```
escribir('x);
resultado = raiz_cuadrada(x);
espacio = espacio_libre() + 512;
valor=raiz_cuadrada(convertir('z',caracter_2));
```

Ejemplos de llamadas a funciones

Otro ejemplo de la flexibilidad que caracteriza a C es que las funciones se pueden invocar desde casi cualquier lugar.

Puede usarse una función sola, como instrucción aparte finalizada con punto y coma, desechándose en ese caso el valor devuelto.

Puede usarse dentro de una expresión aritmética o lógica, sustituyéndose entonces por el valor devuelto.

Puede usarse, incluso, como parámetro a pasar a otra función, o incluso a sí misma. Las combinaciones son casi tantas como podamos imaginarnos.

El próximo mes continuaremos con las funciones y la forma de usarlas en lenguaje C. J.A.B.P.

# Lenguaje Ensamblador Motorola 68000

Retomamos el curso con la última instrucción de la primera parte de nuestro pequeño listado:

**addq.l #4,sp**

Necesitamos ajustar el puntero o registro A7 para devolverle a su condición original para que pueda ser utilizado nuevamente. "add" quiere decir añadir. La "q" junto a "add" significa quick (rápido) y puede ser utilizado con cantidades entre 0 y 7 ambos inclusive.

Hagamos ahora un pequeño paréntesis y aprovechemos el momento para aclarar unos fundamentos imprescindibles para programar correctamente en ensamblador.

Para entender lo que es una posición de memoria tenemos que intentar imaginar como si la memoria fuese una estantería de muchos pisos. En cada piso de la estantería caben , como máximo , ocho libros. Algunos pisos no tienen fondo y tienen acceso a la habitación de al lado. Si colocamos algún libro caen a la habitación de al lado y otros pisos que no tienen libros podemos colocar libros desde la habitación adyacente. Cada ocho bits (libros) forman un grupo denominado byte.

Los pisos normales se denominan bytes de memoria RAM, los que no tienen fondo y caen a la otra habitación los llamamos bytes de acceso a periféricos. Si ponemos libros y caen a la habitación de al lado son posición es de memoria de escritura a periféricos, si los libros entran por el agujero se denominan posiciones de memoria de lectura de periféricos.

Para tener los libros organizados numeramos los pisos y así el primer piso de ellos es el 0, el segundo 1,... . El número de posición de memoria se denomina dirección.

Otro tema a veces difícil de entender es la numeración. Los humanos solemos trabajar en el sistema decimal, probablemente por que tenemos diez dedos. El ordenador , al ser un sistema electrónico, sólo tiene dos estados (dedos ?). Simplificándolo, en una pista del circuito o pin de un chip sólo pueden ocurrir dos cosas : que haya tensión (5V) o que no lo haya (0V). Represen-

tándolo numéricamente que está 1 ó a 0.

Este sistema de numeración la conocemos por "Sistema Binario". Toda la información que circula en el interior de un ordenador evidentemente son unos y/o ceros (5 V ó 0 V). Los primeros programadores de ordenadores debían introducir toda la información a base de unos y ceros. Este sistema binario es real y puramente código máquina aunque durante el curso utilizaremos este concepto como sinónimo de lenguaje ensamblador.

Aunque la primera impresión es de desagrado, las operaciones se realizan más fácilmente en binario que en el sistema decimal.

Cada número se puede descomponer. Cada signo ve multiplicado su valor por una potencia de la base del sistema (o número de símbolos distintos), por ejemplo:

$$524=5 \times 10^2 + 2 \times 10^1 + 4$$
(sistema decimal)

Si hacemos los mismo con un número binario:

$$10011 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 = 19$$

De forma análoga, para obtener la representación binaria de un número decimal se divide sucesivamente por dos, y los restos de las divisiones en orden inverso constituyen sus cifras:

1 9/2

1 9/2

1 4/2

0 2/2

0 1

11001 leído al revés es 10011.

Operar en un sistema u otro se rige siempre bajo las mismas normas.

Por ejemplo si deseamos multiplicar 1001 x 110 en binario deberemos hacerlo así:

```
 1001
 x110
 ____
 0000
  1001
   1001
 ______
 110110
```

Como ejercicio os recomiendo los siguientes :

1) Teniendo en cuenta que el Bus de datos de un ordenador es de 8bit...

¿Cuántos estados distintos pueden representarse ?

2) Escribir en binario 1991.

3) Pasar a decimal:
1001110011.

Como podréis observar el representar un número alto en binario requiere un número de dígitos altísimo lo cual representa una incomodidad y sobre todo induce a cometer errores. Imaginémonos que tenemos que teclear rápidamente el siguiente número :

11110010010000100110
11100110011000110011
00111001000010001111
10100010110011010011

¿Verdad que es fácil colocar un cero en lugar de un uno o viceversa?

Para evitar estos errores los programadores se decidieron primero por el sistema octal y posteriormente , con la aparición de los ordenadores de 16 bit , el sistema hexadecimal.

En el próximo capítulo explicaremos por encima el sistema hexadecimal y hacer una confesión quizá sorprendente pasaremos este tema que reconozco es muy aburrido pero creo necesario que el programador conozca estos conceptos fundamentales.

Por este motivo os recomiendo que compréis uno de los libros de referencia que indico abajo. De esta forma, acompañaréis el estudio de mis artículos con el de estos buenos manuales de referencia.

En el próximo número de Código Juego vamos a tener, por fin, más espacio para hacer nuestros primeros programas consistentes.

Es una promesa. JMR.

---

Bibliografia y teléfonos de interés:

*Guia del programador Tomo I*
(Ibercomp 971-45 66 42)
*Introducing ATARI ST machine code*
(Ed.zzSoft. BitVision 96-586 34 15)
*Devpac 3*
(HiSoft. Isis Informática 98-57 19 17)

**Juan M. Rial**

# CODIGO JUEGO

te lo pone más fácil

Una simple llamada

al 956-840246

y estarás inscrito

para el concurso

EL ROMPECABEZAS INFORMATICO

## LA PREGUNTA DE ESTE MÉS

¿Cuantos bits hay en un nibble?

# ATARI Falcon 030

Habremos oído hablar de él, sabemos que es un potentísimo ordenador. Pero... ¿nos compraremos uno?...

Investigan el tema FALCON; José Manuel Ginés y Juan manuel Rial.

El ATARI FALCON 030, A 32BITs es un potente multimedia. No es una mejora de sus antecesores STE o TT030, por ser éste un diseño completamente innovador y que ha sido creado para llegar a ser el centro de atención de todos aquellos que aman los gráficos y el sonido.

## PRESENTACION

El FALCON viene en una caja de cartón junto a un manual en castellano bastante bueno, un disco con utilidades, otro con el MULTITOS (multitarea), y otros dos con el SpeedoGDOS y el Calapart (Agenda, calendario). El ratón, por supuesto, tambien viene incluido.
Lo que puede sorprender a algún que otro usuario de PC compatibles es que no vienen disquetes con el sistema operativo. En los ordenadores Atari el sistema operativo viene en ROM.
Poner a trabajar el FALCON es tremendamente sencillo. Basta conectar un monitor SVGA o RGB, conectar el ordenador a la red y encenderlo. Después de unos segundos aparece el Desktop (pantalla de trabajo). Esta última pertenece a un entorno gráfico que tambien existe para MS-DOS, llamado GEM de Digital Research.
A diferencia de versiones anteriores la presentación se ha mejorado mucho. Las ventanas , botones y cajas de diálogo aparecen en tres dimensiones y los iconos tienen un acabado muy llamativo. El número de estos es elevado, pero existe la posibilidad de que el usuario añada iconos personalizados.
La novedad principal de esta versión de GEM es la posiblilidad de configurar el idioma y el teclado. En el disco 1 hay un programa llamado CONFIG donde podemos elejir el idioma alemán. inglés, francés o castellano, idem ocurre con el teclado. Cualquier variación se almacena en la RAM estática y al resetear el equipo aparece el idioma que hemos seleccionado con anterioridad.
Puertos E/S: el fabricante Atari siempre ha provisto sus ordenadores de un interface Midi y el FALCON no es ninguna excepción. También ha mantenido, en esta ordenador, los conectores del ratón y joystick en el mismo lugar, debajo del teclado.
A la izquierda del teclado podemos encontrar otros dos conectores extendidos para joystick (como el que viene en la consola Jaguar), el conector de cartucho Rom y los anteriormente mencionados Puertos Midi.
La disquetera de 1.44Mb se encuentra a la derecha (opcionalmente ED) y todos los demás conectores se encuentran en la parte posterior. Empezando por la derecha el primer puerto es un conector DSP, a continuación la salida estéreo, entrada estéreo, puerto SCSI 2, monitor, RF (TV, Pal, NTSC, puedes configurar), centronics, RS232 y puerto LAN.

## SUPERIOR SONIDO

El sonido sampler es muy popular en los ATARIs ST, usando las salidas del cartucho AVR, para crear con ellos samplers mono y estéreo, en una resolución de 8 y 16 bits, casi la calidad del CD (disco compacto) 44.1 KHz.
El FALCON O3O, no necesita ningún cartucho externo, para crear esto. Utiliza para ello un chip especial llamado CODEC (COder-DECoder). El chip CODEC contiene 16 bits ADC y DACs en el mismo ordenador. Lo único que hay que hacer es coger un sonido estéreo e introducirlo en el FALCON através de la entrada de micrófono que incorpora el ordenador en su parte posterior y arrancar el programa de sampler correspondiente. Una vez realizado esto, podrás samplear sonidos, voces, etc... a una

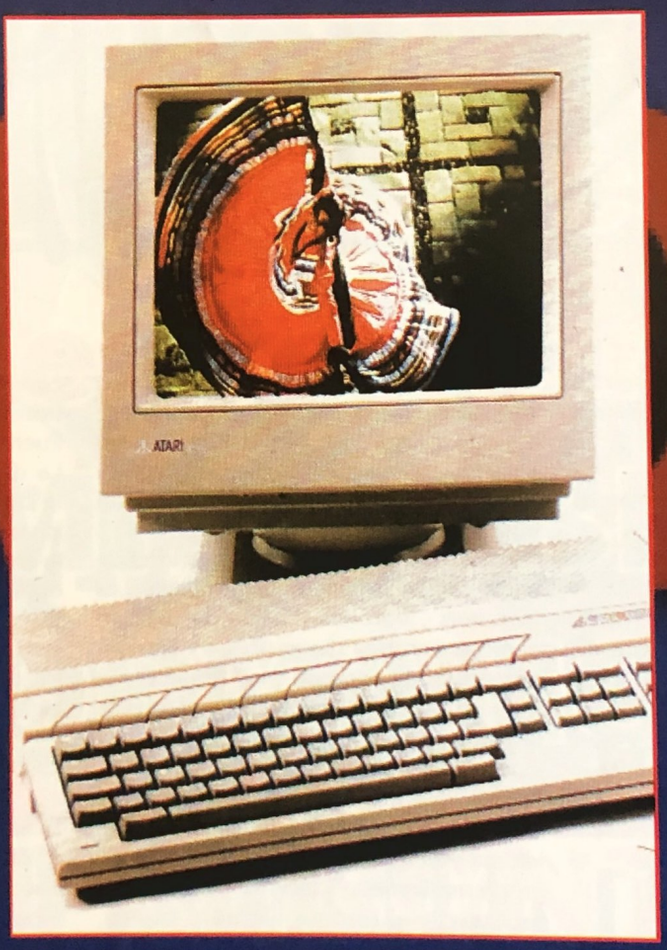
# Analizando e

calidad superior a la del CD, en este caso 50KHz.
Esta función, la realiza directo a disco duro y reproduce la grabación con una resolución de 16 bits, característica innovativa y principal de este ordenador. Además, debemos resaltar que no todos los ordenadores son capaces de realizar funciones semejantes para la industria musical.
Otra parte que integra el subsistema de audio del FALCON, es el Motorola DSP 56001 (Procesador de Señales Digitales), un poderoso co-procesador diseñado para trabajar con la información digital de audio y los efectos visuales, siempre a alta velocidad y en tiempo real. La combinación del CODEC y el DSP, con accesorios del sistema DMA del FALCON, dan a éste el más poderoso sistema de sonido disponible en un ordenador personal. Además, en la famosa Feria de Muestras de NAMM, aseguran que será el líder en el terreno musical por muchos años.

EXPANSIONES DEL DSP

El DSP, puede ser usado para más aplicaciones diferentes que sólo para manipular sonidos, pues puede procesar cualquier tipo de señal digital a alta velocidad y de forma directa al disco duro. El puerto DSP, en la parte posterior del ordenador, es también un interface rapidísimo para modems, fax, escáner color, impresoras laser, genlock y grabación DAT.
El poderoso DSP puede ser usado conjuntamente con el chip CODEC, para reconocimiento de voz, correo por voz y aplicaciones gráficas, las cuales son comprimidas con métodos JPEG y MPEG para películas de vídeo, morphing y cualquier otro efecto visual. Algunos programas existentes para PC, con el formato de vídeo MPEG, han sido mejorados para usarlos con el FALCON.
En el FALCON, el DSP viene provisto de su propia memoria y podemos adquirir un paquete de desarrollo con su ensamblador, debugger, etc... Por ejemplo, puede reducir el tiempo de un rendering de varias horas a cuestión de minutos gracias a él.

Al igual que intervienen en digitalización de audio, añadiendo efectos en tiempo real, pueden intervenir en el tratamiento de imágenes en operaciones de brillo, filtros, contraste, etc... a velocidades muy superiores de las de la propia CPU.
El FALCON 030 posee un gran número de resoluciones gráficas (nosotros hemos encontrado ya casi 60) tanto a 256 colores como a 65536. El blitter es el que se se encarga de los refrescos de pantalla.
Entre las opciones del Desktop (pantalla de trabajo) se ha incluido la opción (definir video). Con esta opción podemos seleccionar el número de colores, 40 u 80 columnas, entrelazado y las resoluciones compatibles con Atari ST. Siendo éstas distintas dependiendo de si el monitor conectado es un monitor VGA o RGB.
Mediante programación se puede acceder a otro gran número de resoluciones, entre ellas con Overscan, resolución muy útil cuando trabajamos con Genlock, ya que no existen los bordes negros que suelen tener las otras resoluciones.
La que suele utilizarse en la mayoria de los casos es 640*480 con 256 colores. Obteniendo una pantalla muy estable, ideal para cualquier aplicación. Para Autoedición podemos adquirir, opcionalmente, tarjetas gráficas con resoluciones hasta 1280*1024. Hemos testeado el Calamus S 2.0 y el Calamus SL del 4/8/93 conjuntamente con una laser HP de 600 dpi y el resultado ha sido francamente bueno. El programa es genial en el FALCON y poder trabajar con imagenes de 24 bits ha resultado altamente placentero.




OTRAS VENTAJAS

No tan obvio, pero casi tan importante, son los otros aspectos del hardware del FALCON 030. El procesador central es un Motorola 68030, que corre a una velocidad de 16MHz. Este procesador es muy rápido si lo comparamos con la velocidad del procesador de otras plataformas.
Hemos de tener en cuenta que el FALCON posee dos CPU s diferentes y cuando ambas trabajan al unísono el ordenador es brutal.
La otra CPU es el DSP56001, que corre a una velocidad de 32MHz. Para los que les suena poco el tema, queremos aclararles una pequeña confusión que existe entre muchos usuarios de ordenadores. Los MHz no indican velocidad, sino, utilizando un símil, expresan las revoluciones por minuto a las que trabaja el computador. Con un motor más potente podemos alcanzar mayor velocidad a menos revoluciones. De todas formas, para aquellos ésto no les convence existen aceleradores de 32/40 MHz y tarjetas con la CPU 68040 a 66 MHz.
El 68030 tiene la capacidad de protección de memoria y memoria virtual. La protección de memoria utilizada en las operaciones del sistema del, es usada para tener más de un programa corriendo a la vez en el sistema y permite una comunicación entre ambos al mismo tiempo (tiempo real).
La memoria Virtual es utilizada en el FALCON 030, para usar la memoria del disco duro como RAM. Si tienes un FALCON 030 de 4Mb de RAM y un disco duro,

# Falcon 030

puedes asignar 12Mbytes del éste como memoria Virtual. Así, de esta forma, tendrás un ordenador de 14Mb, de los cuales 2Mbyte serán usados para las operaciones del sistema.
Hay varias configuraciones del FALCON 030 en el mercado. La primera sin disco duro y con 1Mb de RAM, la segunda con 4Mb de RAM y sin disco duro, la tercera tiene 4Mb de RAM y 86 o 64 Mb de disco duro, la cuarta con 12Mb de RAM y 86 o 64 Mb de disco duro y la quinta, y ultima, con 16Mb de RAM y con 86 o 64Mb de disco duro.
Las versiones que os recomiendo, son a partir de la tercera, debido a que muchos de los programas existentes para el FALCON, son sólo soportados si se dispone el disco duro. El disco duro de fábrica es un IDE (interno), pero podemos poner hasta 5 SCSI (externos) a la vez.
Otra implementación importante en el FALCON 030, es el interface SCSI2, con su controladora, incluida de fábrica. Este es un puerto paralelo de alta velocidad, para conectar todos los periféricos SCSI y SCSI2, como, por ejemplo, el estándar CD ROM, el Kodak foto CD, lectores y escritores ópticos, discos duros extra, etc...
El uso de este último puerto, junto con el sonido y su capacidad grafica, hacen de este ordenador, el más potente multimedia en relación precio-calidad.
Otra cosa a destacar es su velocidad, para lo cual no guiarnos de la de los PCs, debido a que la ingeniería utilizada es más potente que la de los anteriores. Hay que verlo para creerlo.
El FALCON es brillante por muchas razones, y se pueden hacer muchas cosas con él. Pero.... ¿para qué queremos el ordenador? ¿a qué nos dedicamos, o queremos dedicarnos?. Estas son las preguntas que tenemos que hacernos antes de comprar un ordenador.
Lo que más os puede ayudar a este respecto es conocer algo del software existente en el mercado para el FALCON 030. Este apartado se irá implementando mes a mes, debido a la cantidad de programas que ya hay y las pocas páginas de que disponemos en este apartado.
Este ordenador, es uno de los más jóvenes que existen en el mercado, y como tal las casas de software se han volcado con el FALCON, y no paran de sacar cosas nuevas, utilizando todo su poder.
Los modos gráficos del FALCON, pueden usarse con un monitor color RGB, pero si lo que de verdad quieres, es una alta resolución, te puedes comprar un SVGA estándar.
A continuación comentaré, por encima, algunos programas, que en breve espacio de tiempo, os desarrollaré más a fondo.
El primero que vamos a tratar es el TRUE PAINT. Potente paquete gráfico, que trabaja en todas las resoluciones del FALCON, y que pertenece a la casa Hisoft. Puedes crear dibujos, retocarlos, hacer degradados, máscaras, importar gráficos de otros programas, etc.
Lexicor Sofware, la compañía Americana, responsable de los programas gráficos para los ordenadores Silicon Graphics, Iris Indigo y las estaciones de trabajo que crearon los efectos de Morphing en las peliculas Terminator 2 y Abyss, son los responsables de PRISM PAINT 2. Ete programa de gráficos, funciona también en todas las resoluciones gráficas del FALCON, y tiene su explendor en la resolución de True Color (color real). Posee , también un paquete de animación llamado Artis Animator, con nuevos formatos de animación que soportan modos graficos de 24 bit, entre otros.

FOTO STUDIO de la casa Compo, es otro buen ejemplo de un paquete de proceso de imágenes. Utiliza graficos de 24 bits, y 16.7 millones de colores. Importa muchoa tipos de formatos gráficos de toda clase de ordenadores, haciéndolo compatible con los demás.
Los gráficos son cargados dentro de una ventana estándar del GEM del ordenador y accedemos a las funciones del proceso de imágenes desde la barra de menu.

El FALCON, es particularmente fuerte, en Render y Morphing. Render es un proceso de imágenes matemático usado para crear objetos reales en la pantalla del ordenador. Es decir, mediante luces, sombras y la misma escenografía se crea un ambiente casi natural. Un efecto real, podríamos decir que es un espacio virtual, donde nuestros sentidos, crean algo vivo y tridimensional en la fría pantalla del ordenador.
MORPHING, es una técnica para crear animaciones, donde un objeto es transformado en otro en sucesivos pasos siendo este proceso casi imperceptible cuando se realiza paso a paso.
Hay tres paquetes de RENDER en nuestro mercado. El primero es RAYSTART, posiblemente os pase una demo de este programa, así que atentos a Código Juego. Se puede usar sin Coprocesador Matemático, pero si no queréis estar esperando durante horas y horas, habrá que ponérselo al FALCON. Tampoco es tan caro, creo que cuesta unas 10.000 pts. Este programa funciona tambien en todas las resoluciones gráficas del FALCON.

XENOMORPH 2: este programa tiene otra versión más antigua, pero ha sido sustituida por XENOMORPH 2, que trabaja bajo el sistema Multitos y saca todo el rendimiento al FALCON 030. Además, su fabricante ha incluido, en esta última versión, más texturas, efectos, y hasta sonidos samplers.

INSHAPE, es un poderoso y tremendo programa, incluye unas excelentes especificaciones, muchas clases de texturas, efectos de sombras, niebla, temperatura de color, etc...

MULTIMEDIA Y VIDEO PROFESIONAL

El FALCON no es sólo capaz de producir gráficos, es también capaz de abrir muchas y variadas posibilidades de aplicaciones Multimedia. El requerimiento básico para que un ordenador produzca buenos vídeos, es un procesador rápido así como una amplia paleta de colores. Bajo estas consideraciones vemos que el FALCON es ideal para este tipo de aplicaciones.
El FALCON, es capaz de digitalizar imágenes en tiempo real y de grabar éstas directamente a disco. Tambien, tenemos la posibilidad de usar algun programa de Overscan y utilizarlo con Genlock, para presentaciones gráficas de vídeo, cine, etc...
Un ejemplo de este tipo de programas es el VIDEOMASTER, entre otros, que realiza numerosos efectos en tus vídeos. Tiene un cartucho que se coloca en el puerto del cartucho del FALCON, y te permite conectar la camara o cualquier aparato de vídeo. Puedes manipular los Frames, cortar, pegar, hacer Morphing en plan Terminator, etc... Como puedes comprobar todo el buen software de este ordenador se orienta hacia los efectos gráficos, pero todavía hay más.

Continua en página 66

# Microsoft

# MACRO

# ASSEMBLER

## La mejor herramienta

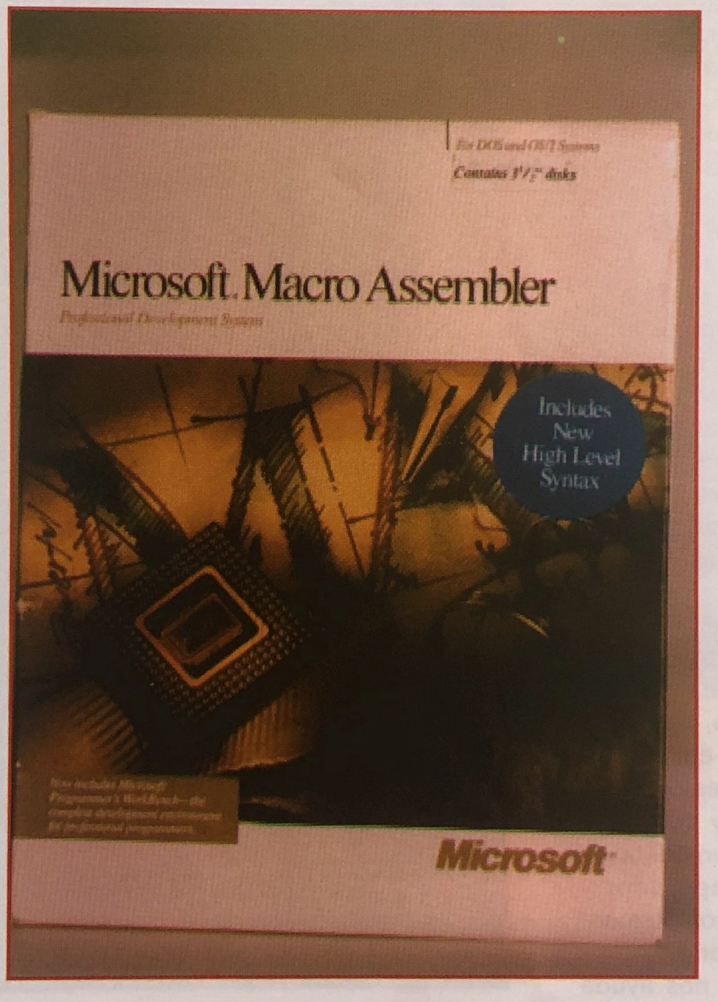
La principal herramienta de trabajo que todo programador debe poseer es el compilador del lenguaje que utilice. En el caso de los que gustamos del lenguaje ensamblador nuestra necesidad se orienta hacia un, valga la redundancia, buen ensamblador, o sea, el programa que nos permite interpretar nuestros nmemónicos y traducirlos en código ejecutable. Sin el transcurso de éste, nuestro programa sólo seria un montón de líneas sin sentido en formato ASCII que no valen para nada, salvo para ocupar espacio en el disco duro.

En Código Juego queremos ayudaros en la elección del ensamblador que más os guste y más se ajuste a vuestras necesidades de programación. Por esta razón, comenzamos una serie de artículos sobre los que están disponibles en el mercado, explicando sus pros y sus contras, a la vez que os adentramos en el maravilloso y potente mundo del rey de los lenguajes.

La compañía nortemericana Microsoft, como todos sabemos, no sólo ha comercializado aplicaciones y sistemas operativos de gran repercusión, sino que también, a la par que lanzaba estos últimos productos, creaba y distribuía ayudas para la programación. Lenguajes como el C, Pascal y Ensamblador, se pueden contar entre los que la omnipotente compañía de software ha implementado y mejorado a a través de los años. El que nos interesa en este caso fue bautizado como MASM, abreviatura de Microsoft Assembler.

PARA LOS QUE SABEN MENOS

El ensamblador es el programa que traduce las órdenes escritas en ASSEMBLY a sus correspondientes códigos o números que el microprocesador puede ejecutar. Los números son difíciles de recordar y es fácil cometer errores al utilizarlos, por lo que en lugar de programar con ellos directamente, le damos a cada uno un nombre genérico y distintivo que nos ayudará a recordarlos siempre que queramos realizar una operación determinada y nos facilitará enormemente la programación, porque lo hacemos en un lenguaje cercano al humano, en este caso, el inglés. Así tenemos órdenes como ADD (sumar), SUB (restar) y MOV (mover). Además, el nombre de la orden no se elije al azar sino que siempre guarda una relación de significado con la tarea que la propia instrucción es capaz de llevar a cabo. Estos como mnemónicos u opcódigos, son los que el programa ensamblador se encarga de traducir en las cadenas de código máquina que el microprocesador es capaz de ejecutar. Esta es , resumida, la función principal del ensamblador, y dotados de ella y poco más nacieron los primeros ensambladores comerciales.




A medida que la programación fue creciendo en posibilidades también lo hicieron las capacidades que se esperaban de los ensambladores, por lo que las compañías creadoras, como Microsoft, se dedicaron a implementar sus productos de manera gradual por medio de las sucesivas versiones que iban viendo la luz a través de los años.

Cuando se usa el lenguaje ensamblador se logran programas puros, que son los que poseen un código totalmente optimizado, es decir, que todas las instrucciones que se utilizan son justamente las que se deben usar y en el formato más adecuado posible para realizar la función necesitada. Para que entendáis esto mejor , pensad que un compilador de C, que también genera código máquina, siempre va a necesitar más cantidad de instrucciones y menos optimizadas para realizar una tarea, que si la hiciésemos en lenguaje ensamblador. El compilador genera un código menos optimizado ya que se crea, podíamos decir, "electrónicamente" y no con el transcurso de la inteligencia humana, al contrario que los programas en código máquina creados y pensados en lenguaje ensamblador.

MASM

Cuando hacemos un programa en lenguaje ensamblador, componemos un listado con un determinado número de líneas. Este se forma de un cierto número de instrucciones en inglés o palabras cortas (de unas 2 a 5 letras) que simbolizan, en su significado, la acción u operación que van a realizar. Utilizamos operaciones básicas como sumar, restar, mover, almacenar, comparar, saltar, que una vez conjuntadas permiten realizar cualquier tipo de operación.

El programa ha de escribirse en formato ASCII o sólo texto sin caracteres especiales de control, lo que también se conoce como texto puro. Este listado se llama FUENTE, ya que es que el assembler va ha tomar como origen o fuente para realizar el proceso. Por lo general a este archivo

se le dará la extensión .ASM (Assembler), aunque puede utilizarse cualquier otra.

MASM utiliza este archivo como entrada y lo convierte en un archivo .OBJ que será utilizado por otra utilidad, también de MICROSOFT, llamada LINK. LINK enlazará el código objeto (.OBJ) y dará como resultado nuestro programa ejecutable (.EXE). Otra utilidad más, EXE2BIN, convierte los .EXE en .COM o imágenes de memoria. Esto último es materia que se trata muchísimo mejor en el curso de ensamblador.

## LA LINEA DE COMANDOS Y EL ENTORNO INTEGRADO

Existen 2 formas de trabajar con un ensamblador. La primera sería aquella a la que todos los que trabajamos con el MSDOS estamos acostumbrados, es decir la línea de comandos. Las diferentes opciones de ensamblado, el nombre del listado fuente y el nombre del fichero de código objeto resultante se introducen en una sóla y compleja línea desde la que el propio programa ensamblador las irá tomando a medida que las vaya necesitando.

Este sistema es mucho más críptico, pero fue el que se tuvo que utilizar durante muchos años. Concretamente desde MASM 1.0 hasta MASM 5.1.

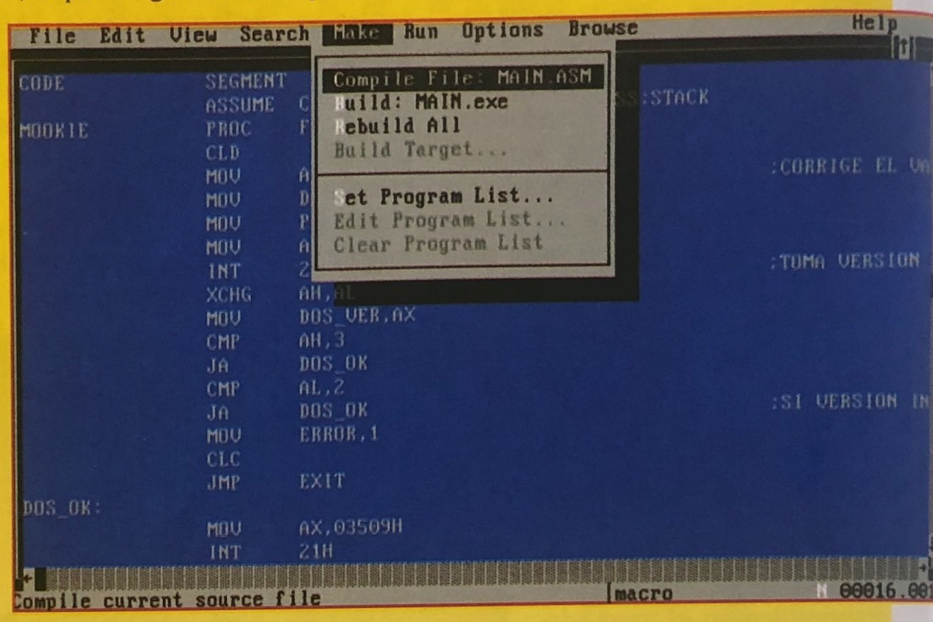
Con la llegada de MASM 6.0 y MASM 6.1 la cosa cambió. Ahora todo el proceso, desde que se escribe la primera línea de código, hasta que el programa corre veloz en nuestro ordenador libre de errores, se halla integrado en una sóla y completa aplicación, que engloba un editor de texto, ensamblador, depurador y otras muchas utilidades complementarias. De esta manera, arrancamos un único programa que nos ayuda enormemente a trabajar, porque integra todo lo que necesitamos en un único entorno.

## PWB

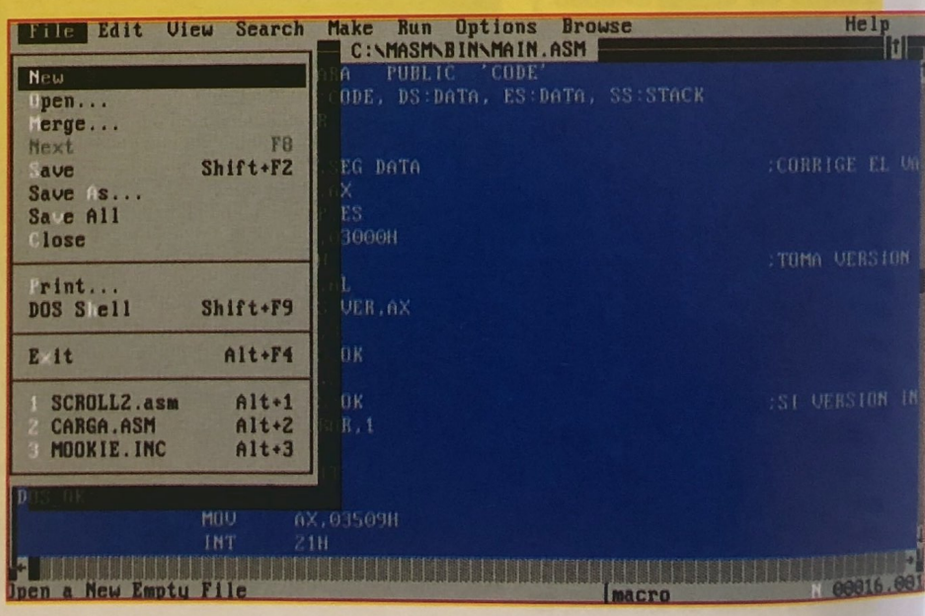
Las siglas PWB significan Programmers' Work Bench, o sea, Banco de Trabajo del Programador. El nombre le viene que ni al pelo, porque es en realidad una mesa de trabajo en la que disponemos de todos los elementos necesarios para una buena programación en ensamblador.

Desde los menus de ventana de PWB accedemos al MASM, LINK, LIB, EXE2BIN, BROWSE, NMAKE, CodeView, etc., con un sólo golpe de ratón.

A diferencia de la línea de comandos, aquí elegimos las opciones




"pinchándolas" o seleccionándolas dentro de las cajas que aparecen




dentro de los diferentes menús PULL-DOWN o descendentes, que cuelgan de la barra superior de la ventana de PWB.

El editor de texto ocupa la ventana principal pudiéndose escribir en ella como el si se tratase de EDIT o cualquier otro al que estamos acostumbrados a ver y utilizar en nuestros ordenadores. Se puede cortar texto, pegar, copiar y buscar, cambiar, etc., como en cualquier otro editor.

PWB es multiventana, lo cual es otra gran ventaja ya que podemos tener más de un archivo abierto a la vez y saltar de uno a otro a nuestro antojo, cortando , copiando y pegando datos entre ellos.

Cuando nuestro trabajo está finalizado ordenamos directamente una CONSTRUCCION o BUILD que engloba todo el proceso de ensamblado y enlazado. Si ocurren errores, y lo más normal es que así sea, se nos abre una ventana de compilación en la que podemos ver todos los que ocurrieron con su tipo y colocación lineal dentro del listado fuente. En ese momento podemos pasar al depurador CodeView que nos ayudará a encontrar todos los BUGS que hemos cometido. Al salir de CodeView regresamos al entorno PWB, el cual no abandonamos hasta

que terminamos nuestra tarea o cuando abrimos una ventana al DOS.

Cuando la compilación o construcción de nuestro programa es correcta se puede hacer que funcione desde el mismo PWB.

BUILD

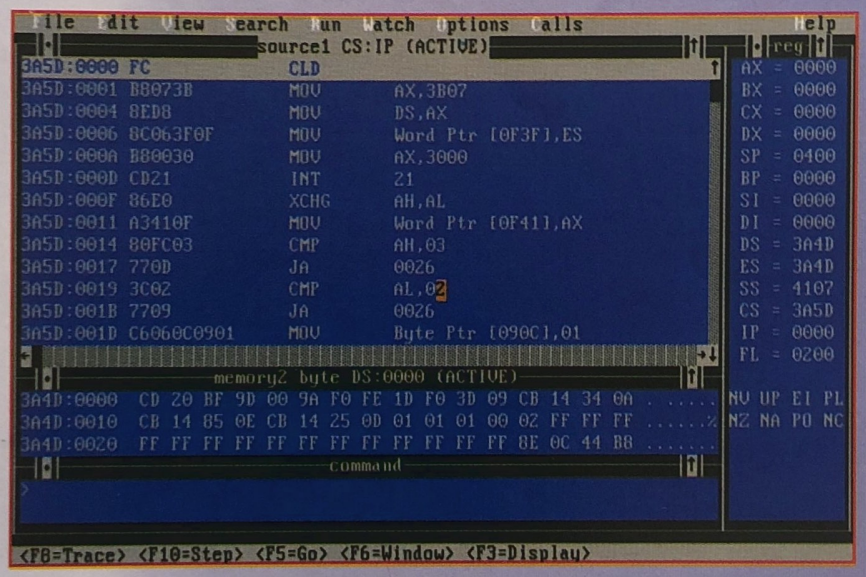
La CONSTRUCCION o compilación completa de nuestros programas es otra característica nueva con PWB y por lo tanto con MASM 6.0. Gracias a esta opción no tenemos que realizar todo el proceso de compilación en varios pasos como antiguamente. Simplemente indicaremos cuales son las opciones que deseamos en cada una de los menús pulldown y a continuación ordenaremos una CONSTRUCCION de nuestro programa. Entonces, se realiza un ensamblado con MASM, un enlazado con LINK (con las librerías opcionales que indiquemos), así como cualquier otro paso que queramos, como EXE2BIN si el programa final tiene el formato COM, añadir datos para CodeView, BROWSE, etc.

ADVISOR ON LINE HELP

Otra característica que surge con MASM 6.0 y 6.1 es la ayuda en línea. Con esta impresionante base de datos que siempre está disponible desde cualquier parte de PWB, encontramos el apoyo necesario en caso de cualquier tipo de duda. Por ejemplo, si no sabemos cual es la sintaxis de una instrucción o no sabemos que significa un determinado error de compilación. Bastará con marcar la palabra o hacer un doble clic sobre ella para que se nos abra un ventana del Advisor On Line Help que nos recordará todo aquello que no sabemos o sobre lo que dudamos.

Como podéis comprobar se trata de un método de auxilio de gran valor puesto que no tendremos que usar el manual tan a menudo y encima no hay que abandonar el entorno de trabajo para resolver este tipo de problemas.

Continuaremos en CJ3 el análisis de MASM. Una vez acabado éste, comenzaremos a utilizarlo para que empecéis a ver las directivas nuevas que se pueden utilizar con este soberbio programa, las cuales convierten a un lenguaje como el ensamblador, en un sistema de programación parecido al de los lenguajes de alto nivel como el C o el Pascal. J.L.




**EL DEPURADOR CODIGO CODEVIEW, PERMITE MEJORAR NUESTROS PROGRAMAS, ASI COMO LIMPIARLOS DE IMPUREZAS. SE TRATA DE UNA APLICACION PROFESIONAL EN CONSONANCIA CON UN PAQUETE DE ALTA CALIDAD**

# HERRAMIENTAS LOS DEBUGGERS

En el anterior capítulo de nuestro curso sobre los depuradores de código, empezamos a estudiar como se le puede sacar partido a éstos para ver los registros del microprocesador, desensamblar código y observar la memoria. En esta entrega vamos a estudiar un comando más de DEBUG que nos permitirá ensamblar pequeños programas y estudiar como funcionan las instrucciones.

El debugger estándar de Microsoft, es considerado como un programa poco profesional y que es incluido en el MSDOS, sea cual sea su versión, un poco por continuidad, porque en realidad es de poca o nula utilidad. Nada más lejos de la realidad. Cuando entramos en un programa profesional de depuración, como puedan ser CodeView, Masterkey, Turbo Debugger, etc., es cierto que se notan unas diferencias muy palpables, pero, a veces, esta potencia de trabajo hay que pagarla en dos puntos vitales. Uno de ellos es la velocidad, ya que en muchas ocasiones compilar un pequeño trabajo puede llevarnos un gran rato mirando, como tontos, la pantalla de nuestro PC. Por otro lado está la complejidad, porque son tantas las opciones de ensamblado que para arrancar este proceso, deberemos configurar exaustivamente nuestro programa. Esto último está muy bien cuando se trata de un programa grande, algo con muchas variables, saltos, etiquetas, etc. Pero, ¿qué ocurre cuando lo que queremos hacer es algo directo, rápido, sencillo y corto?. Entonces se hecha de menos ese pequeño ensamblador/desensamblador que se arranca muy veloz, actua muy rápido y de forma directa, de acuerdo con lo que necesitamos en ese momento. Por ello, vuelvo a insistir, en que no se debe desmerecer y mucho menos olvidar, a esta pequeña utilidad, que encima, todos tenemos en nuestro subdirectorio del MSDOS.

## DEBUG y los REGISTROS

A los que seguís el curso del ensamblador, ya no os sonará extraño el nombre y significado de los registros. La CPU, al realizar sus cálculos, utiliza estos almacenes de memoria temporales de 16 bits (2 bytes), a los que podemos llamar las manos del procesador. Para ejecutar una instrucción, la CPU siempre, o casi siempre, necesita el transcurso de los registros como lugar intermedio para llevar a cabo la operación. Los microprocesadores Intel del la familia 80x86 tienen 2 formas de trabajar, lo cual influye también en la manera en que operan los registros. El modo normal de operación es llamado MODO REAL, que es en el que la CPU trabaja por defecto. El otro, denominado MODO PROTEGIDO, es el que permite a las CPU 80286 en adelante trabajar con el ordenador sin la barrera del MSDOS y su segmentación de memoria y los famosos 640k. Hasta que no sepamos como opera el ordenador en modo real, el modo protegido es prohibitivo por su complejidad. Así que nos centraremos en el primero de ellos.

El microprocesador cuenta con 4 registros de uso general de 16 bits (AX,BX,CX,DX) que se pueden usar como si fuesen de 8 bits, dividiéndolos en 2 partes de 8 bits que se llaman High y Low (H y L) o Parte alta del registro y parte baja del registro. Así, tenemos AH, AL, BH, BL, CH, CL, DH y DL. Las instrucciones son también consecuentes con el tipo de registro que usemos. Los demás registros (segmento, pila, puntero, índice y flags) también son de 16 bits, pero no se pueden dividir en 2 de 8 bits. Cuando ponemos la orden R(Register), DEBUG nos muestra todos los registros incluido el de Banderas o Flags. Al iniciar DEBUG, ya sea mediante el método uno o dos que vimos el mes pasado, los registros contienen una información que puede llegar a ser interesante para el programador.

## LOS FLAGS

Las banderas o FLAGS son de una importancia enorme, ya que la "inteligencia" del microprocesador depende de ellas. Cuando, por ejemplo, la CPU compara dos números, lo que en realidad hace es realizar una resta, sólo que no guarda el resultado en ningún sitio. Tan sólo cambia el contenido de la bandera en cuestión. Después de esta operación, cuando el flag de ZERO ha cambiado, podemos bifurcar gracias a algunas instrucciones que comprobarán el estado de estas banderas y ejecutarán o no su cometido, dependiendo de la situación en que se encuentren estas últimas. A cada bandera le podían haber asignado un byte para guardar su contenido, lo que ocurre es que sólo tienen dos posiciones 1 ó 0, por lo que decidieron meterlos a todos dentro de un mismo doble byte (word o palabra) y ahorrar espacio. Cada bit tiene un significado dentro del registro del flags y sus estados, como bits que son, pueden ser 1 ó 0, verdadero o falso. En el curso de ensamblador se detalla, el estudio de estos. Aquí sólo los vamos a detallar, con el nombre que les asigna DEBUG en sus dos estados.

## ENSAMBLANDO

La verdadera utilidad de DEBUG empieza cuando tenemos que ensamblar. El comando que inicia este proceso es A, abreviatura para ASSEMBLE. La sintaxis de la orden es: A[dirección]. La dirección indicará la posición de memoria a partir de la cual queremos comenzar a ensamblar. Si se omite la dirección, DEBUG empezará a ensamblar a partir de estas dos posibles direcciones alternativas: 1) La última dirección donde se dejó de ensamblar con el último comando A o 2) Si no se usó A previamente, se comienza en la dirección indicada por CS:IP.

Cuando introducimos la letra A, con o sin el argumento de dirección nos aparece en la pantalla el cursor parpadeante para que empecemos a introducir los mnemónicos en ensamblador, situado este último justo a la derecha de la dirección de memoria segmentada a partir de la cual van a introducirse el código máquina derivado del ensamblaje estas instrucciones. Siempre, tras

Continua en página 57

**MUCHOS DE VOSOTROS OS HABREIS PREGUNTADO COMO ES QUE SI DESDE EL PRIMER NUMERO DE CODIGO JUEGO ENSEÑABAMOS ENSAMBLADOR, C, Y MUSICA NO DEDICABAMOS ESPACIO PARA LOS SONIDOS DE NUESTROS JUEGOS. PUES BIEN, SUBSANAMOS EL ERROR CON ESTE CURSO QUE OS TRAERA TODOS LOS MESES UN MONTON DE COSAS SOBRE ESTE IMPORTANTISIMO ASPECTO DE LA PROGRAMACIÓN DE LOS ORDENADORES HOY EN DIA.**

**¡¡OS VAIS A CHUPAR LOS DEDOS !!**

# NUEVO CURSO

# LOS MIL Y UN SONIDOS DEL PC

**Roberto Cruz**

## El sonido en la programación de Juegos

Un juego debe tener buen argumento , buenos gráficos, una acción trepidante y un buen sonido. Normalmente este último punto se descuida bastante y encontramos juegos con unos componentes muy buenos pero en los que la acción se ve deslucida por un sonido muy deficiente.

Para solventar este fallo que solemos encontrar en los juegos, os proponemos este curso mediante el cual podréis mejorar vuestros juegos en un mil por cien.

Normalmente se añaden pequeños sonidos a los juegos que en ocasiones nos vemos en la necesidad de desconectar, si el programador almenos nos deja esa posibilidad. También debemos tener en cuenta que hasta hace muy poco tiempo los ordenadores PC no disponían de capacidad para reproducir más alla de unos sonidos muy simples. Hoy la

situación ha cambiado, la proliferación de tarjetas de sonido nos permite una calidad nada desdeñable, dando éste un nuevo sentido a los juegos de ordenador.
Para empezar veremos que es una muestra digital de sonido y como poder utilizarla.
Aunque no seamos grandes músicos todos nosotros podemos crear juegos con una calidad muy buena de sonido, simplemente deberemos digitalizar un sonido analógico y pasarlo a formato digital. Me diréis esto es sencillo pero, ¿Cómo?. Un sonido es una combinación de impulsos eléctricos que cambian de intensidad para reproducir las diferentes frecuencias através de un altavoz. Estos cambios no nos son útiles en nuestro ordenador, porque este tipo de señal es una señal analógica y lo que pretendemos es traducirla al lenguaje del ordenador que es el binario o por ejemplo 00011101. Para realizar esta transformación las tarjetas de sonido cuentan con un decodificador modular de pulsos o conversor analógico-digital que se encarga de analizar los diferentes valores de entrada y traducirlos a sus iguales binarios.
Existe otro factor importante como es la calidad. La calidad de digitalización depende de tres factores que son :

1.- La Frecuencia. Se mide en Khz (kiloHertzios) y depende directamentedel segundo punto.

2.- La Resolución: es el modo en que almacena el sonido digitalizado y puede ser en :

8 bits: 256 posibles valores
16 bits:65536 valores

3.- Si el sonido es mono o estéreo: el modo estéreo consume el doble de memoria pero separa el sonido en dos canales diferenciados.

El sonido así digitalizado se almacena en un fichero. Existen varios formatos pero empezaremos explicando uno de los más habituales como es el del PROTRACKER con extensión .MOD, para lo cual adjunto el siguiente listado para que empecéis a investigar.

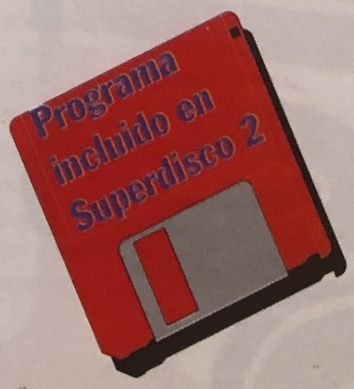
# Archivos MOD

```
#include <stdio.h>
#include <stdlib.h>

/* Definición de la estructura del fichero sample.*/

        struct SampleInfo
        {
        char Name[22];
        int Len;
        short FineTune;
        short Vol;
        int RepStart;
        int RepLen;
        } Sample[31];

        short SongLen;
        short Dummy;
        int SongPos[127];
        char ModSig[5];

        char Use[]="\nUso :  modinfo <modnombre.mod>\n"
                        "Este programa de da información \n"
                        "sobre un fichero .mod \n"
                        "Escrito por BrainStorm & Captain Sarcastic.\n"
                        "kkoller@nyx.cs.du.edu\n";

        char Head[]="\nModInfo 1.9 by BrainStorm & Captain Sarcastic\n"
                          "Genera informacion de  ProTracker Module \n"
                          "Este programa es Freeware\n";

        char Protracker[]="ProTracker 1.1";

int main(int argc,char *argv[])
{
        FILE *fp;
        int outloop,loop,er;
        char temphi,templo;
        char modname[21];
        int RepStart, RepLen;
        char ModType[20];

        if (argc < 2)
        {
        fprintf(stderr,"%s",Use);
        return(5);
        }
```

```
        fprintf(stdout,"%s",Head);

        fp = fopen(argv[1],"rb");

        if (fp==NULL)
          {
          fprintf(stderr,"\nFichero <%s> no encontrado, o error de
fichero.",argv[1]);
          return(5);
          } /* endif */

        for(loop=0;loop<20;loop++)
         {  modname[loop] = fgetc(fp); }
         modname[20] = '\0';   ='\0'  o tambien 0 (cero)

        for (outloop=0;outloop<31;outloop++)
        {
          for(loop=0;loop<22;loop++)
           {
           Sample[outloop].Name[loop] = fgetc(fp);
           }
           temphi=fgetc(fp);templo=fgetc(fp);
           Sample[outloop].Len=temphi*256+templo;
           Sample[outloop].FineTune = fgetc(fp);
           Sample[outloop].Vol = fgetc(fp);
           temphi=fgetc(fp);templo=fgetc(fp);
           Sample[outloop].RepStart = temphi*256+templo;
           temphi=fgetc(fp);templo=fgetc(fp);
           Sample[outloop].RepLen = temphi*256+templo;
        }
        SongLen = fgetc(fp);
        Dummy = fgetc(fp);
        for(outloop=0;outloop<128;outloop++)
          {
          SongPos[outloop] = fgetc(fp);
          } /* end for */
        for(outloop=0;outloop<4;outloop++)
          {
          ModSig[outloop]=fgetc(fp);
          } /* end for */
          ModSig[4]=0;

        fclose(fp);

        if ( (ModSig[0]='M')&&(ModSig[2]='K') )
         {
         strcpy(ModType,Protracker);
         }
        else
         {
         fprintf(stderr,"\n<%s>  no es un  protracker
module.\n",argv[1]);
         return(5);
         }

        fprintf(stdout,"\nModule nombre  = \"%s\"\n",modname);
        fprintf(stdout,"Module tipo  = %s\n",ModType);
        fprintf(stdout,"Song longitud  = %d (%02x
hex)\n",SongLen,SongLen);
        fprintf(stdout,"Muestras  =");
        for (outloop=0;outloop<SongLen;outloop+=8)
          {
          for (loop=0;loop<8;loop++)
           {
           if ( (outloop+loop) < SongLen)
           fprintf(stdout," %02x",SongPos[outloop+loop]);
           }
          fprintf(stdout,"\n              ");
          }

        for (loop=0;loop<31;loop++)
        {
          fprintf(stdout,"\nSample %d ",loop+1);
          if (Sample[loop].Len=0)
           { fprintf(stdout,"- empty");
             fprintf(stdout,"\nText   : \"%s\"\n",Sample[loop].Name);
           } /* endif */
          else
           {
           fprintf(stdout,"\nNombre   : \"%s\"\n",Sample[loop].Name);
           fprintf(stdout,"Longitud : %d bytes (%04x hex)\n"
            ,Sample[loop].Len, Sample[loop].Len);
           fprintf(stdout,"Volumen : %d (%02x hex)\n"
            ,Sample[loop].Vol, Sample[loop].Vol);
           RepStart = Sample[loop].RepStart;
           RepLen   = Sample[loop].RepLen;

           if (! ( (RepStart=0) && ((RepLen=1)||(RepLen=0)) ) )
            {
             fprintf(stdout,"Información de Duración\n");
             fprintf(stdout,"Empieza  : %d (%04x
hex)\n",RepStart,RepStart);
             fprintf(stdout,"Termina : %d (%04x hex)\n",RepLen
,RepLen  );
            } /* endif */
           } /* end else */
        } /* end for (loop++) */

return(0);
} /* end main() */
```

Buena investigación y hasta el mes que viene...

BrainStorm



# Mucho por poco dinero

# POWER PC

## IBM, Apple, Motorola

Siendo la informática un campo tan variable, donde todo los elementos fluctuaban tanto en precio como en prestaciones, no es extraño que los que pertenecemos a él, hayamos desarrollado un instinto natural para dirigir nuestro dinero hacia el lugar correcto. Por ello, cuando algo nuevo sale a la luz, todos nos quedamos expectantes hasta que vemos que se hunde o sale a flote definitivamente. Entonces se desatan las dudas más horripilantes; ¿vendo el que tengo y me compro ese?, ¿Cuánto me darán por el mío?, ¿Necesito, realmente, este producto?. Y es que la envidia nos corroe cuando vemos que nuestro mejor amigo se compró eso último de lo último que nos quita el sueño. Pues bien, vuelve a repetirse la historia de nuevo. En esta ocasión su nombre y apellidos son POWER PC.

Aunque todos sabemos que siempre se le adjudicaron las siglas PC al ordenador que originalmente creó IBM, en realidad estas abrevian el término Personal Computer u Ordenador personal, que se puede utilizar con cualquier tipo de computador, independientemente de su procedencia o marca.

Así, el Macintosh de Apple, Amiga de Commodore, Falcon de Atari y el PS/1 de IBM son todos PCs. Debemos, pues, hacer que la denominación pierda esas connotaciones que el gigante azul siempre quiso adjudicarle. Lo anterior, aunque parezca una tontería escrita en tinta, es un tema que va a cobrar una gran fuerza, con la llegada la poderosa novedad que estará muy pronto nuestras tiendas preferidas.

La idea surge de parte de 3 potentes compañías del mundo informático: IBM, Apple y Motorola, las cuales han decidido unir sus esfuerzos, en lugar de seguir peleandose, para crear un microprocesador con el nombre de POWER PC (Potente PC), que, como dijimos, auna la denominación de PC para estas tres famosas marcas de fabricantes de ordenadores y componentes electrónicos.

¿El porqué de la unión?, tal vez el hecho de que no hay que ser adivinos para darse cuenta de que la idea de ir cada uno por su lado está caduca y llamada a arruinar la informática con el tiempo. El usuario, finalmente, acabaría harto de la incompatibilidad entre productos y decidiría no mejorar su sistema.

Otra razón importante es que estamos en un cruce de caminos. Al igual que Windows hizo que el 286 pareciese una calculadora de juguete, el Multimedia esta convirtiendo los 66 Mhz en una tortuga coja. Las aplicaciones comen ciclos de reloj como churros y en 8 megas de RAM apenas si cabe un procesador de texto de los de la última generación.

Ahora que llegó la voz, el sonido, la música y los gráficos nos percatamos de que hacen falta mejores ordenadores para que estos medios de comunicación sean utilizados másivamente.

El consorcio IBM, Motorola y Apple se dio cuenta de esto y sentados juntos en una mesa decidieron que lo primero que había que hacer era crear un microprocesador de diferente formato y filosofía que los que INTEL estaba haciendo. Por aquel entonces, ya se estaba desarrollando una nueva manera de ver las cosas llamada RISC, la cual fue tomada por la nueva empresa y llevada hasta sus últimas consecuencias.

### UN NUEVO MICROPROCESADOR

El pilar básico de los Power Pc's es el microprocesador del mismo nombre. Se trata de una torre de Babel que nace y crece alrededor del rey de los chips. Las siglas POWER, a parte de significar **poder,** cuando el térmi-

no es leído sin diseccionarlo, significan Performance Optimization With Enhanced RISC. En el lenguaje de Cervantes, tan rebuscada palabreja viene a decir algo así como Optimización del rendimiento gracias a la mejora del RISC. Muchos os preguntaréis ¿qué es eso del RISC?, pues vamos a explicarlo un poco para que todos sean capaces de entenderlo.

Si leéis el artículo sobre el curso de ensamblador de este mes, podréis encontrar allí los fundamentos básicos de como funciona un microprocesador. De todas formas, os diré el proceso por encima: Un procesador normal pide de la memoria la instrucción a ejecutar. Cuando la recibe, la decodifica, la ejecuta y guarda el resultado. A continuación, y esto es lo importante, toma otra de la memoria y la ejecuta después, y así sucesiva e incansablemente. Las instrucciones con las que cuenta para realizar las operaciones, el juego de instrucciones, son de tipo específico y complejo.

Por ejemplo, existen instrucciones que realizan multiplicaciones, divisiones, rotaciones, etc. Estas, toman multitud de ciclos de reloj ya que, aunque parece que es sólo una operación, en realidad el microprocesador tiene que romperlas en muchas pequeñas operaciones. Las instrucciones de este tipo se conocen como complejas y de ahí el nombre de CISC que toman las CPU´s que siguen esta norma de proceso.

Los micros de INTEL, nuestra querida familia 80x86, son de este tipo. Lo que ocurrió es que un día, los diseñadores de microprocesadores empezaron a pensar en que muchas de las instrucciones que ejecuta la CPU son desglosables en otras más pequeñas. Potenciando la velocidad de estas instrucciones más simples, a la par que se hacía un juego (o SET) de instrucciones menos complejo los ordenadores podrían trabajar muchísimo más rápido. La intención era, por lo tanto, REDUCIR el numero de instrucciones. De ahí nace el nombre RISC, Reduced Instruction Set Computer u Ordenador de Juego de Instrucciones Reducido, en contra del CISC, Complex Instruction Set Computer, Ordenador de Juego de Instrucciones Complejo.

**EL PRIMER SISTEMA DISPONIBLE ES EL POWER MACINTOSH DE APPLE.**

Apple

Aunque la tecnología RISC se empezó a gestar tiempo atrás, se dio a conocer en Octubre de 1992, tras que IBM y Apple se uniesen para fabricar conjuntamente con Motorola, el primer PowerPC llamado PPC 601 o MPC 601, dependiendo del nombre del fabricante (IBM o Motorola). Ya se les conoce como PC's de altó rendimiento, y si las expectativas se cumplen, la informática tal y como la conocemos va a cambiar.

Los ordenadores estarán dotados de reconocimiento de voz, dibujo tridimensional y renderización de imágenes en tiempo real, un sueño donde el multimedia y la realidad virtual van a hacer estragos.

## EL USUARIO SE BENEFICIA

Tres cosas hay que resaltar de la aparición del PPC. En primer lugar la estupenda política de mercado que el consorcio fabricante se ha planteado. A fin de introducir el PPC másivamente en el mercado, conscientes de que así recibirá el apoyo de software que le dará definitivamente el empujón hacia arriba, las licencias para la introducción del PPC en placas de terceros fabricantes serán concedidas libremente. Esto traerá consigo que el PPC se extienda en la misma proporción que cualquier microprocesador de INTEL y con ello el boom está casi ganado. En segundo lugar, los precios del PPC han sido ajustados a la baja para que todo usuario pueda acceder a él. Esta maniobra de mercado se debe a que la licencia del engendro pertenece a más de una compañía, con lo que entre ellas mismas se prevé una sana competencia que siempre mantendrá los precios lejos de la manos monopolizantes, como es el caso de Intel Corporation. En tercer lugar, y lo más importante, el usuario final y comprador se verá beneficiado con un ordenador de ensueño a un precio genial y encima apoyado por un software abundante y de calidad.

A esta imparable carrera ya se han unido las empresas creadoras de sistemas operativos. Así, el Windows NT y OS/2 van a ser reescritos para usar el modo nativo del nuevo procesador y los más comunes, como

el DOS y el Windows, podrán correr mediante un emulador, como los de las compañías Insignia Solutions (SoftWindows) y WABI. Se abren unas amplias puertas en el futuro de la informática moderna y el usuario espera impaciente a que así sea.

## El primer procesador de la gama Power es el PPC o MPC 601

### ¡VAYA CISCo!

Retomemos la parte que nos interesa a los programadores, es decir, la naturaleza del funcionamiento interno del propio procesador PPC 601. La tecnología Pipeline, escalar y superscalar no nacen con el Power, ya tienen una aplicación anterior en los micros de la familia Intel. De hecho, los procesadores 486 y Pentium presentan algunas de las características que encontramos en los RISC, por lo que las diferencias no son tan abismales entre unos y otros. El problema de INTEL es que siempre miró atrás al crear algo nuevo, para mantener la compatibilidad con la enorme cantidad de software que existía. De hecho el Power PC 601 tiene la misma velocidad de proceso que el Pentium pero realiza una mejor ejecución de las instrucciones, un mejor manejo de la memoria y las entradas y salidas gracias a su MMU, Memory Management Unit, y una ejecución de las instrucciones más lógica gracias a los 3 canales de ejecución diferentes con los que cuenta. El principal problema con el que se encuentra un microprocesador de este tipo es el hecho de que otra compañía ya es dueña y señora del mercado. Ahora deben luchar contra un software y hardware que apoyan desde hace años a las plataformas PC convencionales. Por lo tanto, para que un usuario, decida vender su sistema y adquirir un PowerPC éste ha de ofrecer algo más que el hecho de ser igual, o un poco más, veloz que un Pentium de Intel. Y de hecho es así. Por un lado tenemos una máquina capaz de correr la inmensa mayoría del software que existe actualmente y por otro, en su modo de trabajo nativo, es capaz de procesar hasta tres instrucciones a la vez mediante un sistema de procesamiento denominado PIPELINE. Este consiste en la implementación de varios canales de proceso diferentes que hacen que se puedan ejecutar como máximo ese número de instrucciones por ciclo de reloj. El RISC y los sistemas escalares y superescalares están llamados ser los dueños de la informática de sobremesa e incluso los ordenadores de mayor potencia.

En definitiva, la arquitectura RISC es el futuro de la filosofía de construcción de los microprocesadores que cada vez necesitan más cantidad de MIPS para mantenernos contentos.

Además, la estandarización que el PPC traerá consigo, presumiblemente, hará que todos los que tenemos un PC, sea de clase que sea, podamos mirarnos a la cara sin tirarnos la incompatibilidad a la cabeza. Tenga éxito o no, lo que si está claro es que es una buena tentativa de progreso.

El usuario, ahora, puede hacer poco más o menos lo que comentaba al principio; esperar atento a que el engendro crezca y se afiance en el mercado o que no levante cabeza, cosa que preveemos improbable, y acabe guardado en el amplio sótano de la informática inútil, a la que ya hemos enviado alguna otra cosilla. Si tiene éxito, cualquier día de estos, cuando menos no lo esperemos, al llegar a la casa de un amigo, exclamaremos ¡Qué maravilla de gráficosi, y él contestará orgulloso, "Es un PowerPCi. En ese momento comenzarán las cábalas y las cuentas de lo que tenemos en el banco, porque no podremos aguantarnos sin tener uno.

Deseemos lo mejor al PPC para que, algún día, todos los informáticos, vayamos juntos, todos a una, cogidos de la mano de la compatibilidad.

IBM, Apple, Motorola

# PowerPC

Viene de pag. 50

introducir la orden deberás pulsar ENTER. DEBUG realiza entonces un chequeo de sintaxis de la orden. Ver figura.

Cuando los datos son traducidos, se introducen en memoria los dígitos en código máquina resultado de esta interpretación.

Si a continuación, hacemos un desensamblado, deberán aparecer las instrucciones y el código máquina como podéis ver en la figura.

Hay que indicar que DEBUG no tiene las facilidades que ofrece un ensamblador profesional, por lo que la forma en que podemos introducir las sentencias de lenguaje ensamblador difiere un poco de las que, normalmente, utilizaríamos con un programa con más recursos.

La principal diferencia es que DEBUG es un ensamblador de un sólo paso. Esto viene a indicarnos que no podemos utilizar etiquetas para nuestras variables y los puntos de salto o bifurcación, como pueden ser las subrutinas o los procedimientos.

Esta traba hace que se tengan que calcular a mano el número de bytes que queremos saltar hacia delante o hacia atrás.

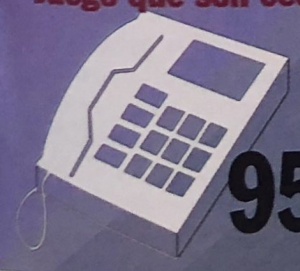
Tampoco se pueden utilizar números decimales o binarios, ya que DEBUG presupone que todos son hexadecimales por defecto.

El mes que viene continuaremos con esta sección que sirve como complemento del curso de ensamblador. Continuaremos explicando como debemos ensamblar correctamente con DEBUG, para que nuestros programas funcionen correctamente.

```
C:\>DEBUG
-A 0100
19C6:0100 MOV AX,DS
19C6:0102 MOV BX,AX
19C6:0104 MOV [0200],AX
19C6:0107 MOV DS,EX
                  ^ Error
19C6:0107

-U 0100 L10
19C6:0100 8CD8        MOV     AX,DS
19C6:0102 89C3        MOV     BX,AX
19C6:0104 A30002      MOV     [0200],AX
19C6:0107 207061      AND     [BX+SI+61],DH
19C6:010A 7274        JB      0180
19C6:010C 69          DB      69
19C6:010D 63          DB      63
19C6:010E 756C        JNZ     017C

-DEBUG BUSCA LOS ERRORES DE SINTAXIS ANTES DE ENSAMBLAR -------
```


# YA ESTÁ AQUÍ EL SUPERDISCO DE CODIGO JUEGO 2

El superdisco contiene utilidades creadas por los programadores de la revista así como programas SHAREWARE y de DOMINIO PUBLICO que han sido seleccionados por expertos en la materia para que os puedan ayudar en la programación de vídeojuegos. El Shareware requiere un pago posterior al creador original del programa en cuestión, no así los que son propios de la revista Código Juego que son cedidos gratuitamente a sus lectores.



956-840246

El precio del disco es de 950 pts, gastos de envío NO incluidos.

PÍDELO A ESTE NÚMERO DE TELEFONO. PUEDES PAGARLO POR TRANSFERENCIA, GIRO, TALON O CONTRAREEMBOLSO.

# El MS DOS y el programador de videojuegos (II)

**Como estuvimos viendo en el anterior número de Código Juego, el MS-DOS permite que otros programas de aplicación puedan correr en nuestro querido PC, ocupandose de todos las funciones importantes dentro de la anatomía del ordenador por lo que, sin su auxilio, nuestros programas no sabrían como realizar muchas de las tareas más básicas. En este artículo, nos centramos en como el sistema operativo es capaz de tomar las riendas del ordenador desde un principio, cuando lo encendemos.**

## ARRANCA EL MOTOR

Para entender que es lo que sucede dentro de nuestro ordenador, no hay nada como empezar a tirar del hilo de los complicados procesos que tienen lugar en él desde su inicialización, o sea cuando pulsamos el botón de encendido que es justo cuando esta enmarañada madeja empieza a deshacerse.

Gracias al proceso de arranque, el ordenador se va convirtiendo, progresivamente, en un "ser inteligente". Al principio, cuando la sangre de la electricidad no corre por sus venas, nuestro PC no es sino una colección de componentes electrónicos muertos: las tarjetas de vídeo, controladoras de discos, los propios discos, la CPU, la memoria ROM y RAM, etc, todos permanecen inertes. Sin esos pocos voltios que lo hacen despertar, el ordenador no realiza ninguna función útil, permaneciendo en el más silencioso de los descansos.

Pero entonces llegamos nosotros, los impacientes usuarios ávidos de juegos y pulsamos el botoncito de encendido. Y mientras esperamos, venga a salir mensajes en la pantalla y algún pitido de vez en cuando. Nuestro corazón se encoje de miedo, hasta que por fin aparece el inductor del DOS (C:>_) que nos indica que todo a ido bien. Por ello, vamos conocer que es lo que ocurre exactamente, con lo que os aseguro desaparecerán nuestros temores.

## LA CPU

Como el PC siempre ha sido un ejemplo de compatibilidad, en 1981, fecha de su creación, los ingenieros decidieron establecer unas reglas fijas que marcasen , de una forma inequívoca, cual sería el proceso a seguir en el arranque de este tipo de ordenadores. Este sistema constituye, aún, una norma que se continúa en nuestros días y aunque presenta algunas variaciones siempre, en el fondo, es el mismo.

Lo primero que ocurre cuando la electricidad entra en el ordenador es que ésta se dirige hacia el microprocesador. Se produce entonces un proceso inicial que se conoce como RESET, que es el mismo que se origina cuando , voluntariamente, pulsamos el botón con el mismo nombre en la carcasa de nuestro ordenador.

El RESET es una señal eléctrica que recibe el microprocesador en una de sus patillas (recordad que la CPU es un chip con multitud de "patitas" que son sus conexiones con la placa base). Esta señal, que como he mencionado también se produce cuando pulsamos o conmutamos el botón de RESET, genera una serie de acontecimientos que conducen a la limpieza de los registros de trabajo tanto del propio microprocesador como de los chips de apoyo del mismo.

Una vez que concluye, el registro IP del procesador es cargado con un dato que siempre es el mismo, en este caso es F000. IP es el contador de programa, es decir, donde la CPU va apuntando cual es la próxima instrucción que debe ser ejecutada. El procesador actúa siempre en consonancia con lo que este registro le dice, por lo que la instrucción localizada en FFFF:0000 será ejecutada en primer lugar.

El segmento F000 está localizado en ROM. ¿Por qué?, muy sencillo. El microprocesador debe encontrar ese primer programa inmediatamente, cosa que no ocurriría si estuviese en RAM, ya que sabemos que ésta se borra cuando desconectamos el PC de la red. Esta es la punta del hilo del que hablábamos con anterioridad. A partir de la primera instrucción, ver foto en página 61, comienza todo el proceso de inicialización en el PC.

**El proceso de arranque de un PC es breve pero a la vez complejo.**
**Se puede dividir en dos pasos principales que engloban el autotest de los componentes del hardware y la carga y acomodación del sistema operativo en memoria, desde donde pasará a controlar todos los sucesos que ocurren en el ordenador.**

## EL POST

Las siglas POST son acrónimo de Power On Self Test, es decir, Comprobación en el encendido.

Cuando encendemos el PC, todos los componentes que lo integran son, podíamos decir, seres desconocidos. Esto se debe, principalmente, a que las configuraciones que puede llegar a tener nuestro ordenador son infinitas , mientras que el sistema operativo que los va a manejar, es siempre, o casi siempre, el mismo. Debido a esto, debe haber una comprobación de todo aquello que puede estar conectado a nuestro PC estándar.

De esto se ocupa el POST, el cual, mediante un paulatino y gradual chequeo, comprueba los distintos periféricos que han sido incluido en el montaje del PC a la vez que los incluye, de forma lógica, dentro del sistema total.

Lo primero que realiza la CPU es un autochequeo y después hay una comprobación del propio código del POST para asegurarse de que el ordenador no se va a colgar cuando lo ejecuta, ya que podría estar dañado de alguna manera.

Gracias al BUS que es el "autobús" que transporta datos y señales en general, por toda la física del ordenador, el microprocesador se puede comunicar con todos los periféricos. Es en este momento cuando el POST les envía señales para ver si están vivos y coleando.

Otra parte importantísima es el RELOJ del sistema. Para que podáis entender su función deberéis cerrar los ojos y pensar en uno de esos antiguos barcos donde los esclavos remaban. Siempre había una persona que tocaba un tambor y marcaba el ritmo de remado que es precisamente la tarea del RELOJ del PC, el cual hace que todos los componentes actúen de forma sincronizada y precisa. En el siguiente paso del POST, se realiza una comprobación del buen funcionamiento del mismo.

A partir de este momento se estudian los periféricos de forma intensiva, asegurándose que funcionan perfectamente y a continuación se los incluye dentro de el sistema BIOS total del sistema.

Como sabemos, el BIOS se encarga de el manejo básico de las entradas/salidas del PC. Pero también en él se incluyen todas las rutinas (programas) que ofrecen al sistema operativo, que se carga más tarde, para que éste sepa como trabajar con el periférico en cuestión. Supongamos, por un momento, que la tarjeta de vídeo que yo tengo es una TRIDENT, por ejemplo y que mi sistema operativo es el MSDOS en cualquiera de sus versiones. A la hora de sacar algo en la pantalla, el DOS echará mano del BIOS de la tarjeta de vídeo para hacerlo ya que nadie mejor que él sabe como realizar esta tarea a la perfección. Supongamos ahora que no disponemos de este código especializado en la BIOS de nuestra TRIDENT y por lo tanto el programa para realizar esa salida de información por la pantalla, tiene que estar incluida, de forma estándar en el propio sistema operativo. Todo va estupendamente hasta que lo usamos con una tarjeta PARADISE. Al estar diseñado el programa para una Trident no funcionará con la nueva tarjeta ya que no son compatibles. Por lo tanto, para que este problema no surja, lo mejor es dejar que le pro-

pio fabricante de la tarjeta diseñe el programa para trabajar con ella y que este sea incluido en la ROM BIOS de ésta. Estos criterios también se aplican a todos los periféricos que se conectan al PC y que necesitan su propia ROM BIOS para funcionar.

En realidad, lo que se hace en este paso del POST es darle el control al BIOS de las tarjetas controladoras de los distintos elementos (tarjeta controladoras de discos duros y floppies, tarjetas de vídeo, tarjetas de escáneres, tabletas digitalizadoras, tarjetas de redes) y se deja que ellos se autocomprueben.

La primera es la de vídeo, como señalamos antes, y se realiza un test de la señal y de la VRAM o memoria RAM del adaptador. A continuación, por lo general, se saca un mensaje por la pantalla con la marca del fabricante y el modelo de la misma, indicando que todo ha ido bien.

Entonces se pasa al chequeo de la memoria RAM del sistema. El microprocesador envía datos a las distintas celdillas de memoria y después intenta releerlos. Si encuentra algún error el sistema se detendría dando el mensaje que causó el error (Por ejemplo: Memory Parity Error, error de paridad). Algunas placas, como la AMI permiten activar un test todavía más completo, pero que enlentece el arranque.

Entonces, se comprueba el teclado. Vemos como nuestras teclas de Scroll Lock, Num Lock y Caps Lock se encienden y en seguida se apagan.

Las unidades de disco, tan frecuentes y tan variadas ya sea en formato fijo o removible, son comprobadas en el siguiente punto del POST y los resultados se comparan con la tabla de configuración que se guarda en la CMOS, que como sabéis no se pierde cuando se apaga el PC por que está continuamente alimentada por baterías, normalmente de Níquel-Cadmio recargables. Los chips CMOS están especialmente diseñados con unos componentes que les permiten consumir muy poca potencia y por lo tanto permanecer inalterables durante largo periodos de inactividad con el ordenador.

Una vez que todos los componentes de nuestro PC han sido comprobados y sus BIOS han sido chequeadas e incluidas dentro del conjunto lógico del ordenador, el sistema operativo puede entrar en escena desde el disco.

## EL MSDOS, A ESCENA.

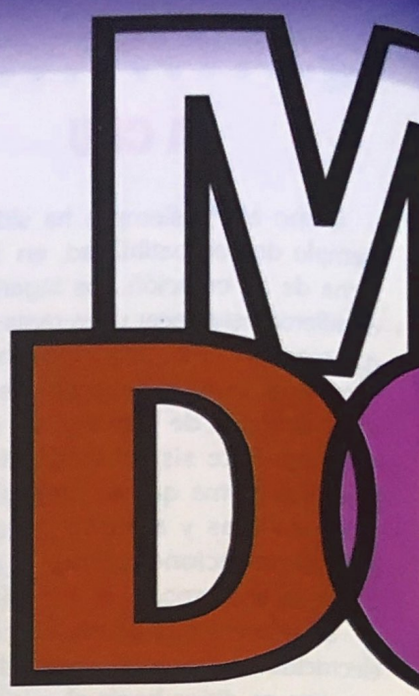
Muchas personas que no entienden el ordenador y mucho menos sus características básicas de funcionamiento, me preguntan por qué un ordenador tan potente como el 486 todavía tiene que andar cargando el sistema operativo desde el disco como el antiquísimo PC original, y qué si no habría un procedimiento más ágil.

Creo que estas mismas cuestiones se os pueden haber planteado también a vosotros, amigos lectores, por lo que paso a contestar ambas.

1) Es cierto que es un engorro y que un ordenador como el 486 merece tener un sistema más avanzado, pero esta forma de funcionamiento esta pensada para beneficio de nosotros, los usuarios. Si es verdad que los discos duros son muy rápidos hoy en día pero aún así, no son todo lo veloces que podría llegar a ser tener el sistema operativo también en ROM, al igual que el BIOS. Pero, por un lado, esto sería más caro y por otro a la hora de incluir una nueva versión del sistema operativo, tendríamos que andar quitando tornillos y chips, cosa que a no todos los usuarios de ordenadores PC les gusta.

2) Un procedimiento más rápido sería, como he dicho, un chip de quita y pon o tal vez los tiros vayan más orientados hacia un disco óptico super-rápido. El caso es que, como los sistemas operativos son variados (MSDOS, DRDOS, UNIX, OS/2) y a la vez de cada uno de ellos hay varias versiones, el soporte donde lo guardemos no puede ser algo fijo o difícil de sustituir.

Por lo tanto, nuestro querido disco duro o el disquete parecen las soluciones más lógicas a nuestro problema, a costa de la lentitud. ¿Qué le vamos a hacer?, habrá que auguantarse un poco.



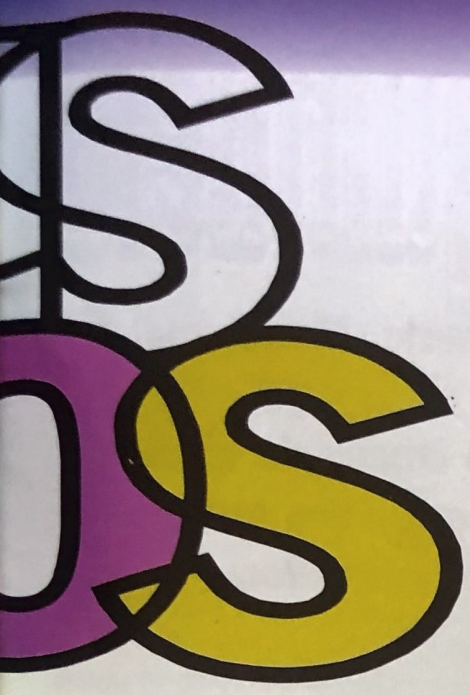
## EL BOOTSTRAP

Así es como se conoce a el programa, que también esta guardado en ROM como el POST, encargado de traer el sistema operativo a la memoria, banco de trabajo de la CPU.

El bootstrap es, pues, un programa auxiliar que sólo se usa una vez durante el proceso de inicialización del ordenador, tras el cual ya no vuelve a ser invocado a no ser, claro está, que se arranque de nuevo el ordenador.

A mí, personalmente, me gustaría en gran medida hablar con alguno de los ingenieros que desarrollaron el PC para preguntarles qué significa exactamente la palabreja *Bootstrap*, por la definición no puede estar más alejada del contexto que pretende, ya que viene a querer decir algo así como "cinta de cuero que sirve como ayuda a la hora de introducirse una bota en el pie". Yo, la verdad, no le veo sentido.

Volviendo al tema, este pequeño programa desempeña un papel importante: busca el sistema operativo que vayamos a utilizar allí donde se encuentre. Al decir cualquier tipo de OS no nos equivocamos, porque no tiene porque ser nuestro querido MSDOS. Con el PC podemos utilizar cualquier sistema operativo, como puede ser el UNIX o el OS/2, siempre que este disponible en el lugar que lo va a buscar el *Bootstrap*. De todos modos, y a partir de ahora, asumimos que nos referimos al más



utilizado de todos.

## BUSCANDO A:>

Cuando todo el proceso de comprobación del hardware a concluido, empieza lo verdaderamente importante.

Ya tenemos un buen montón de componentes electrónicos que sabemos, positivamente que funcionan a la perfección, pero necesitamos el software que lo haga funcionar. El bootstrap siempre intentará cargar el sistema operativo desde la unidad A:. Es en este momento cuando la lucecita de la disquetera se ilumina y al los pocos segundos se apaga.

En ese momento pueden ocurrir dos cosas. Una es que tengáis un disco dentro con lo que estáis indicando al bootstrap que el sistema operativo se encuentra ahí almacenado. Otra es que no haya nada en esa disquetera con lo que se pasa intentar cargarlo desde el disco duro C:. Si no lo encuentra en ninguno de estos dos lugares el bootstrap no da un mensaje en la pantalla indicando el error.

Si, por el contrario, se os olvido quitar un disquete de la unidad A: y éste no contiene el sistema operativo, también obtenemos un error que nos invita a sustituirlo y pulsar una tecla. ¿Cuántas veces no os ha ocurrido esto a vosotros?, cientos de veces diría yo. Pues bien, ahora sabéis la razón de por qué ese error hace que nuestro ordenador no sea capaz de arrancar. Sin el sistema operativo a mano nuestro ordenador no sirve.

```
C:\DOS>DEBUG FORMAT.COM
-U
1556:0100 E93983        JMP     843C
1556:0103 43            INC     BX
1556:0104 6F            DB      6F
1556:0105 6E            DB      6E
1556:0106 7665          JBE     016D
1556:0108 7274          JB      017E
1556:010A 65            DB      65
1556:010B 64            DB      64
1556:010C 0000          ADD     [BX+SI],AL
1556:010E 0000          ADD     [BX+SI],AL
1556:0110 4D            DEC     BP
1556:0111 5A            POP     DX
1556:0112 2401          AND     AL,01
1556:0114 42            INC     DX
1556:0115 0000          ADD     [BX+SI],AL
1556:0117 0020          ADD     [BX+SI],AH
1556:0119 00FA          ADD     DL,BH
1556:011B 00FF          ADD     BH,BH
1556:011D FF05          INC     WORD PTR [DI]
1556:011F 09B00000      OR      [BX+SI+00001],AX
-
```

```
C:\>DEBUG
-U FFFF:0000
FFFF:0000 EAF404CF05    JMP     05CF:
FFFF:0005 3131          XOR     [BX+D
FFFF:0007 2F            DAS
FFFF:0008 3131          XOR     [BX+D
FFFF:000A 2F            DAS
FFFF:000B 3932          CMP     [BP+S
FFFF:000D 00FC          ADD     AH,BH
FFFF:000F 0000          ADD     [BX+S
FFFF:0011 0000          ADD     [BX+S
```

## Los discos

No es este el artículo indicado para describir cual el formato que tienen los discos formateados con el MSDOS, cosa que se estudiará en una próxima y suculenta lección, pero si os vamos a dar una leve reseña.

Los discos se dividen de forma lógica en una serie de partes que permiten al sistema operativo acceder a la información, ya sea para grabarla o recuperarla. Estas partes se denominan pistas, sectores y clusters. Podemos pensar en un disco como en un montón de cajas donde los datos se guardan y se borran sin mayor problema y que están muy bien organizados. La primera caja de todas sería la etiquetada con un 0 y así sucesivamente hasta todas las que nos permita tener la densidad de nuestro disco.

El bootstrap siempre buscará los datos del sistema operativo en la caja 0 de la cara 0. Las caras, como podéis ver, también se numeran de igual forma.

Si no está ahí entonces se entiende que el disquete no contiene el sistema operativo y se produce el error.

## El sector de arranque

En realidad, la función del bootstrap es la de cargar un programa especial que se encarga realmente de traer el sistema operativo a la memoria. Este programa se encuentra, como hemos dicho, en el sector 0 cara 0 del disco donde éste está guardado. El bootstrap lo carga en una dirección arbitraria y le da el mando, finalizando de esta manera su función.

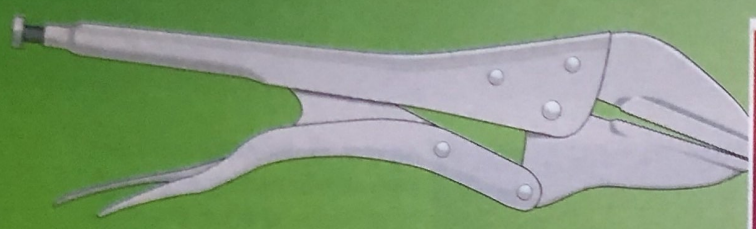
El sector cero, cara cero de un disco que contiene el sistema operativo se denomina SECTOR DE ARRANQUE ó boot sector. El pequeño programa que este sector contiene es cargado por el bootstrap en la memoria RAM en el marco del segmento 7C00 hexadecimal. En este momento, la BIOS finaliza su función y se le da el control al software que no pertenece al Firmware del ordenador.

El sector de arranque copiado en RAM toma el mando y busca, en el mismo disco del que ha venido, a los dos archivos que forman el sistema operativo MSDOS y que se llaman IO.SYS y MSDOS.SYS. De nuevo, si alguno de estos dos importantísimos archivos no pudiesen ser encontrados se generaría el error "No es disco del sistema, sustituyalo y pulse una tecla".

El código del sector de arranque siempre ocupa 512 bytes, que es justamente el tamaño de un sector.

En el próximo artículo veremos como IO.SYS entra en la memoria RAM y como el MSDOS es definitivamente cargado.

Hasta Agosto.
M.A.L.



# La triquiñuela

Letras que suenan a gloria

**Cuantas cosas hace tu ordenador y tu sin saberlo. Menos mal que algunos nos enseñan como.**
**Ya hemos recibido muchas cartas y disquetes trucos de alucine. Este mes hemos seleccionado este, para que veas que de cosas podemos hacer a los mandos de nuestro ordenador.**

SONTEXT v.1.1 Por JULIO RAMOS - Oviedo 1994

¿Qué es SONTEXT?

SONTEXT permite visualizar ficheros de texto en modo teletipo. Producirá un golpe de teclado por cada carácter visualizado.

Notas importantes: Asegurar de no utilizarlo con ficheros de texto de más de 80 columnas; no pasa nada si es usa con este tipo de ficheros, pero la visualización no quedaría bien. La longitud del fichero puede ser de cualquier tamaño.

Modo de funcionamiento

SONTEXT ———— Muestra una pequeña ayuda
SONTEXT ?
SONTEXT /H ———— " " " "
SONTEXT <fichero.ext> [velocidad]

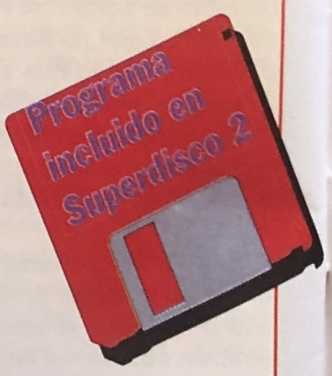
Si le añadimos el parámetro velocidad debe ser en 0 y 10. Si omitimos este parámetro la velocidad es 2.

Ejemplos:

SONTEXT LEEME.DOC
SONTEXT C:\AUTOEXEC.BAT
SONTEXT README.TXT /1

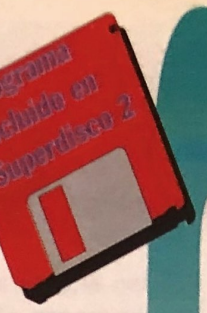
## SONTEXT.C

```
/* SONTEXT —— Teletipo para ficheros de texto
   SONTEXT <fichero.ext> [velocidad]          */

#include <stdio.h>
#include <io.h>
#include <dir.h>
#include <dos.h>
#include <conio.h>
main(int argc, char *argv[])
{
static char caracter;
int vel;
FILE *fich;

if (argc<=1 || strcmp(argv[1],"/?")==0 || strcmp(argv[1],"/H")==0 || strcmp(argv[1],"/h")==0)
  { salir(); exit(0); }

if (argv[2]!='')
   {
   if (atoi(argv[2])<0 || atoi(argv[2])>10) vel=2;
    else vel=atoi(argv[2]);
   }
else
  vel=2;

if((fich=fopen(argv[1],"r"))==NULL) { printf("No existe el fichero: %s",argv[1]); exit(0); }
clrscr();

     cursor(1);
     while (!feof(fich) && !bioskey(1))
                {
                  caracter=getc(fich);
                  if (caracter!=' ' && caracter!='\n')
                   {
                    tiempo(vel);
                    beep();
                   }
                  printf("%c",caracter);

                }

bioskey(0);
cursor(0);
fclose(fich);
}

salir()
{
clrscr();
textcolor(14);
gotoxy(30,1); cprintf("SONTEXT v.1.1");
gotoxy(18,3); cprintf("Programado por JULIO RAMOS - Oviedo 1994");
textcolor(13);
gotoxy(1,6); cprintf("Visualiza un fichero de texto en modo teletipo, ideal para presentaciones.");
gotoxy(1,8); cprintf("SONTEXT <fichero.ext> [velocidad]");
gotoxy(1,10); cprintf("- El nombre de fichero debe ser un fichero de texto.");
gotoxy(1,11); cprintf("- La velocidad debe de ser entre 0 y 10, cuanto más bajo más velocidad.");
gotoxy(1,12);  cprintf("  Por defecto la velocidad es 2.\n\n\n");
textcolor(7);
return(0);
}

beep()
{ int son; int cont=0; son=inp(97); outp(97,79);
while (cont++<5000);
outp(97,son);
}

tiempo(int velocidad)
{
long inicio,final;
inicio=biostime(0,0);
do
  final=biostime(0,0);
  while(final-inicio<velocidad);
  inicio=0; final=0;
}

cursor(valor)  /* 0=Activa, 1=Desactiva  */
int valor;
{
int carac;
int cara;
union REGS var;
if (valor==0) { carac=6;cara=7; }
   else { carac=255; cara=255; }
var.h.ah=0x01;
var.h.ch=carac;
var.h.cl=cara;
int86(0x10,&var,&var);
}
```




**Por Julio Ramos**
**Oviedo**

# LIST.COM
## Muchas cosas en una




Tantos y tantos programas salen al mercado que es difícil saber cuales son los que necesitamos en un momento determinado. Por ello, sin saberlo, tenemos cerca de nosotros pequeñas maravillas que solucionan grandes problemas. Estos programas, apodados DOMINIO PUBLICO o SHAREWARE, han sido creados por gente que ha trabajado mucho, de forma desinteresada en algunos casos y no tanto en otros. Eso sí, los precios que pagamos por este tipo de software nunca son elevados en comparación con la calidad que recibimos a cambio. Además, nos dejan problarlos antes. ¿Qué más se puede pedir?. En esta nueva sección mensual, no ocuparemos de analizar todos los programas y buscar aquellos que puedan servir de utilidad al programador. Hoy, comenzamos con un magnífico ejemplo. Recuerda que el SHAREWARE hay que pagarlo tras un tiempo prudencial de prueba, para ello os damos todos los datos de la empresa y el programador.

### LIST

En muchas ocasiones, he oído el mismo comentario: "El MS-DOS es una porquería. Los archivos no pueden tener más de ocho caracteres en el nombre, no hay ratón, las ordenes no hay quien las entienda, etc.". Que le vamos a hacer, tienen parte de razón. Los que como yo, y probablemente tú, gustan de trabajar con la línea de comandos del MS-DOS, debemos quitarnos el sombrero al sentarnos frente a un Macintosh, por su versalitidad y maniobrabilidad. Esto esta cambiando, habrá dicho más de uno de vosotros al leer el párrafo anterior, pero aún así, todavía queda mucho por andar. Y como todo parece indicar que a nuestro "feo" sistema operativo sigue teniendo mucho que decir, nos vemos obligados a conocerlo y entenderlo. Lo que sí ocurre es que nunca nos hemos conformado, me refiero a los que usamos el MS-DIOS, a aliviarnos la carga de tan críptico sistema operativo. Por esta razón, principalmente, paralelo al desarrollo del DOS han nacido y crecido unas utilidades que a todos nos han hecho la vida más fácil. Los programas a los que me refiero son conocidos por todos; entornos (shells) , desde donde podemos utilizar los comandos y posibilidades del sistema operativo, pero de una forma más racional y cómoda. Habrá cientos de ellos, no es broma. Desde los que son famosos como el Norton Commander y PC Tools hasta los que, como el que comentamos hoy, no lo son tanto, o por lo menos no tan famosos en los cauces comerciales del software. Pero si lo es entre los ratones de BBS, que al igual que yo, disfrutan descubriendo esa pequeña utilidad que hace muchas y buenas cosas.

Los SHELLS, como el que hoy comento, tienen todos una dinámica parecida y ofrecen, mediante el uso de ventanas y ratón, una manera sencilla de Formatear, Crear Directorios, Borrar/Mover/Copiar archivos, Renombrar, Grabar, y un largo etcétera, comandos que si no fuese por programas como LIST tendríamos que teclearlos desde la línea de comandos.

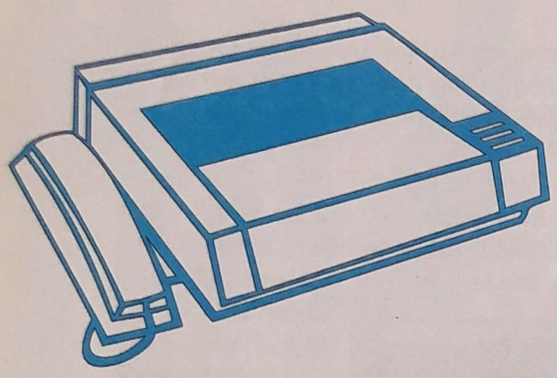
Potencia de LIST

El nombre lo toma de su principal función; listar archivos. Pero, ahí no acaba la cosa, por supuesto. Todavía, después de hacer funcionar LIST en bastantes ocasiones, no puedo creerme como han podido hacer un programa que haga tantísimas cosas y que ocupe sólo 25 kilobytes (versión 7.7a). Es alucinante la rapidez de funcionamiento y posibilidades que nos ofrece este programa, que según su propio autor lo podemos utilizar siempre que no sea para un uso comercial pero si a nivel privado o personal. Debemos, por lo menos, enviarle un regalo de unos 20 dólares, como el mismo indica en una de las pantallas. Nosotros, Código Juego se los vamos a enviar a Vernon, que así se llama el programador, para que veas que te apoyamos en tu magnífico trabajo. Tenemos pues un programa que posee muchísimas posibilidades y por poco o ningún dinero. Entre aquellas que merecen atención especial están: Uso de doble ventana de visualización, listado de archivos en formato ASCII y Hexadecimal, grabación de mediante tecla de opciones de configuración, cambio de colores, búsqueda de cadenas de texto ASCII en el interior de archivos, borrar, copiar, mover, renombrar e imprimir ficheros, etc.

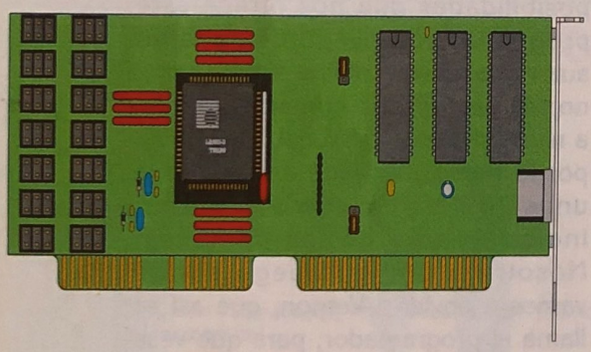
Para un programador, LIST posee muchas ventajas, que a buen seguro os van a hacer enamoraros de él, como tantos y tantos usuarios mundiales. Una recomendación CJ.

```
1                    03/15/94 17:58 • HELLO.OBJ
000000  80 16 00 14 5C 54 43 5C 45 58 41 4D 50 4C 45 53  Ç▬ ¶\TC\EXAMPLES
000010  5C 48 45 4C 4C 4F 2E 43 67 88 1F 00 00 00 1B 54  \HELLO.Cgê▼   ←T
000020  43 38 36 20 42 6F 72 6C 61 6E 64 20 54 75 72 62  C86 Borland Turb
000030  6F 20 43 2B 2B 20 31 2E 30 31 92 88 1C 00 00 E9  o C++ 1.01ÆÊ└  Θ
000040  4C 8F 6F 1C 14 5C 54 43 5C 45 58 41 4D 50 4C 45  LÃo└¶\TC\EXAMPLE
000050  53 5C 48 45 4C 4C 4F 2E 43 0A 88 1D 00 00 E9 20  S\HELLO.C◙ê↔  Θ
000060  08 3A 15 15 43 3A 5C 54 43 5C 49 4E 43 4C 55 44  ◘:§§C:\TC\INCLUD
000070  45 5C 53 54 44 49 4F 2E 48 C1 88 06 00 00 E5 01  E\STDIO.H┴ê♦  σ☺
000080  00 00 8C 88 06 00 00 E5 01 03 00 89 88 05 00 00    îê♦  σ☺♥ ëê♣
000090  E7 0D 00 7F 88 05 00 00 E7 0D 00 7F 88 03 00 00  τ♪ aê♣  τ♪ aê♥
0000A0  E9 8C 88 05 00 00 EA 01 09 7F 96 28 00 00 05 5F  ΘîÊ♣  Ω☺○aû(  ♣_
0000B0  54 45 58 54 04 43 4F 44 45 05 5F 44 41 54 41 04  TEXT♦CODE♣_DATA♦
0000C0  44 41 54 41 04 5F 42 53 53 03 42 53 53 06 44 47  DATA♦_BSS♥BSS♦DG
0000D0  52 4F 55 50 D1 98 07 00 28 0D 00 02 03 01 26 98  ROUP╤ÿ• (♪ ☺♥☺&ÿ
0000E0  07 00 48 0C 00 04 05 01 03 98 07 00 48 00 00 06  • H♀ ♦♣☺♥ÿ• H  ♠
0000F0  07 01 0B 9A 06 00 08 FF 02 FF 03 55 8C 0A 00 07  •◙♂U♦ ◘ ☺ ♥Uî◙ •
000100  5F 70 72 69 6E 74 66 00 71 88 0B 00 00 E3 18 00  _printf qêó  ↑↑
000110  00 00 23 04 00 01 4A 88 04 00 00 E0 18 7C 90 0C    #♦ ☺Jê♦  α↑|É♀
000120  00 00 01 05 5F 6D 61 69 6E 00 00 00 5A 88 0B 00    ♣_main   Zêó
000130  00 E3 19 00 00 00 23 04 00 00 4A 88 05 00 00 E1   ↑↓   #♦  Jê♣  ß
000140  19 18 61 88 15 00 00 E6 06 73 69 7A 65 5F 74 0A  ↓↑aê§  µ♦size_t◙
000150  06 06 66 70 6F 73 5F 74 06 06 3C A0 11 00 01 00  ♦♦fpos_t♦♦<á◄ ►
000160  00 55 8B EC 8B 00 00 50 E8 00 00 59 5D C3 19 9C   Uï═╕  Pè  Y]├↓£
Command▶     *** Top-of-file ***          Keys: ↑↓→← PgUp PgDn  F10=exit F1=Help
```

# El contestador



Volvemos, este mes, a encontrarnos en el apartado de Código Juego donde intentamos solucionar esas pequeñas dudas que os acosan y que, en ocasiones, no os dejan dormir. Ya sabes, si deseas plantearnos tu problema sólo tienes que llamarnos, escribirnos o enviarnos un FAX. (CODIGO JUEGO, Apdo. de correos 114, 11520 Rota (Cádiz), Teléfono 956-840246, FAX 956-813129).



**¿Qué VESA es esa?**

Me llamo Gonzalo y soy un fanático de la informática. Tengo un 486DX2 con 8 megas de RAM y un disco duro de 130 MB (Seagate), que me costó sangre, sudor y lágrimas conseguir, porque no es fácil convencer a Papa de que se gaste tanto dinero. Ante todo, felicitaros por la revista, la cual, yo creo, ha pillado de sorpresa a todo el mundo. Pienso que nadie podía esperar tener la suerte de que la primera publicación de estas características se publicase en España. ¡Felicidades! Bueno, vamos ya con el problema. Desde siempre he querido aprender lenguaje ensamblador, pero nunca he tenido nadie que me echase una mano, por lo que no he llegado a aprenderlo con toda la soltura que quisiera. Sin embargo, lo poco que he aprendido me ha hecho decidirme, finalmente, a estudiarlo a fondo. Y es que este lenguaje me ha permitido hacer cosas increíbles (Pronto os enviaré alguna cosilla para que la evaluéis, a ver qué os parece). En este momento, intento crear un shell para utilidades de disco y quiero darle un aspecto "chuli" con una pantalla a todo color en el formato de vídeo 640x480 . Lo que ocurre es que yo tengo una tarjeta de vídeo TRIDENT 8900c y tengo entendido que si programo código para ella, puede que este no funcione en otras tarjetas como la Paradise, por ejemplo. ¿Es esto cierto?.

Gonzalo Prieto Sanz
Palma de Mallorca

CJ: Gonzalo, es una alegría lo que nos dices y te damos las gracias desde la redacción de Código Juego. Este tipo de cosas nos animan, en gran medida, a seguir esforzándonos en mejorar la revista. Tu pregunta tiene por respuesta un sí rotundo. Al principio, cuando IBM era la que llevaba la voz cantante en todo a lo que compatibilidad PC se refería, todo era más fácil. Seguramente, habrás oído decir que muchos cocineros estropean la sopa. Pués esto es exactamente lo que ocurrió en el caso de las tarjetas de vídeo cuando llegó el momento de pasar de la VGA, bien llamada estándar, a los modos de vídeo Super VGA. Al no existir un sistema universal que los distintos OEMs (Fabricantes de componentes) pudiesen haber copiado y, por tanto, haberse convertido en la norma del mercado, cada uno tiró por su lado. El resultado; un montón de tarjetas, capaces de hacer lo mismo pero de diferente forma. Las tarjetas de vídeo que ves en el mercado tienen dentro un componente principal que las hace únicas. Esta parte del hardware del adaptador se encarga de las funciones principales. Al ser diferente para cada tarjeta, también lo es su funcionamiento y, por lo tanto, la forma en que han de ser programadas. Los problemas del programador son evidentes ya que cada una direcciona y organiza la RAM de vídeo (VRAM) de diferente forma. De esta manera, como tu bien dices, tu TRIDENT, cuando se programa a nivel de ensamblador, y no mediante llamadas al BIOS, dará unos resultados que serán imprevisibles en otra marca de adaptador de vídeo. La solución, al menos parcial, llegó de la mano del comité VESA (Video Electronics Standard Association). Las principales compañías OEM de tarjetas de vídeo se decidieron, finalmente, a reunirse y ponérselo fácil a los programadores mediante la creación de una BIOS de vídeo ampliada que permitiese a los creadores de software, la estandarización de sus programas, en lo que a presentación visual se refiere. Esta BIOS VESA, es simplemente un driver que el fabricante suministra en forma de una archivo .SYS. En el futuro, las nuevas tarjetas ya incluirán este driver en ROM con lo que su carga en memo-

*Cómo pudisteis comprobar en el pasado número de Código Juego, en esta sección nos hacemos eco de vuestros problemas con el PC, que sabemos que son muchos. Si este es tu caso no dudes en enviarnos tu carta o fax con vuestra duda lo mejor documentada posible.*

ria convencional no será necesaria. En el disquete del número anterior incluimos una utilidad de Dominio Público que te permite chequear tu tarjeta de video y saber algunos parámetros importantes. De todos modos, la programación a nivel de hardware, que no de BIOS, se estas tarjetas es todavía un misterio.

### Toda una demostración

En el último CD-ROM que llegó a mis manos, pude comprobar que abundaban las DEMOS de juegos. Mis preguntas son dos: 1) ¿Requieren estas demos una forma especial de programación diferente al programa original o versión comercial? 2) Si esto son DEMOS, ¿Por qué llaman demos también a las que aparecen en los BBS y que no son de juegos?, ¿Cómo las programan?

Manuel Isabal López
Pontevedra

CJ: 1) No, por lo general no. Te diré que, la versión DEMO de un juego es el mismo juego comercial, pero "arreglado" de forma que no puede funcionar completamente. Este arreglo se logra mediante lo que los programadores llamamos "parcheado" del inglés PATCH. Una vez que el juego se completa y se comprueba hasta la saciedad, se decide insertarlo en el mercado. Para ello, muchas compañías creadoras de software optan por crear una versión recortada del mismo para , por así decirlo, permitir al usuario su utilización y después dejarlo con la miel en los labios. El parcheo de un programa puede hacerse de muchas formas. Si el programa esta bien estructurado (Rutina principal-subrutinas) es relativamente sencillo, ya que con el cambio de algunas sentencias en el bucle principal habremos logrado nuestro objetivo. Si, por el contrario, el juego se divide en partes y cada una se guarda en archivos diferentes (del tipo FASE1.DAT, FASE2.DAT, etc.) bastará con no incluir en el paquete de la demo las fases que no deseemos. Nunca un juego se hace pensando en la demo, al igual que nunca una película se hace pensando en el trailer publicitario. Pero, ya te digo, en un 80 por ciento de los casos la versión demo contiene todo el programa ejecutable de la versión comercial. Tan sólo, se le retoca un poco. Cada vez existen más tipos de demos comerciales de juegos. Entre ellas podemos distinguir, a groso modo, las jugables, no jugables y slide-show.

2) Creemos que las demos a las que te refieres son las de los grupos de programadores , por lo general, nórdicos. Estas, son completamente diferentes a la que he explicado arriba, tanto en concepto como en finalidad. Se podría decir, que en función de su nombre, las verdaderas demos son estas, porque son las que demuestran cosas. Estas maravillas de la programación nacen en el seno de los grupos de programadores de otras plataformas, como AMIGA y ATARI, que estában más preparadas para la música y los gráficos y por lo tanto mejores para este tipo de "shows". Se trata de rivalizar con los programadores de otros grupos e incluso de intentar ridiculizarlos, creado rutinas que hacen cosas alucinantes como SCROLL multiplanar y multidireccional, música, efectos de sonido, gráficos 3D, etc. Como el PC se está convirtiendo en un superordenador también estos dos campos en los que siempre anduvo más débil, las demos de grupos de programadores empiezan a florecer por todos sitios. No hacen nada, pero si son dignas de ver. La programación se suele realizar en el más puro ensamblador. Ten en cuenta que algunos gráficos de estas demos llegan a alcanzar una velocidad de movimiento increíbles.

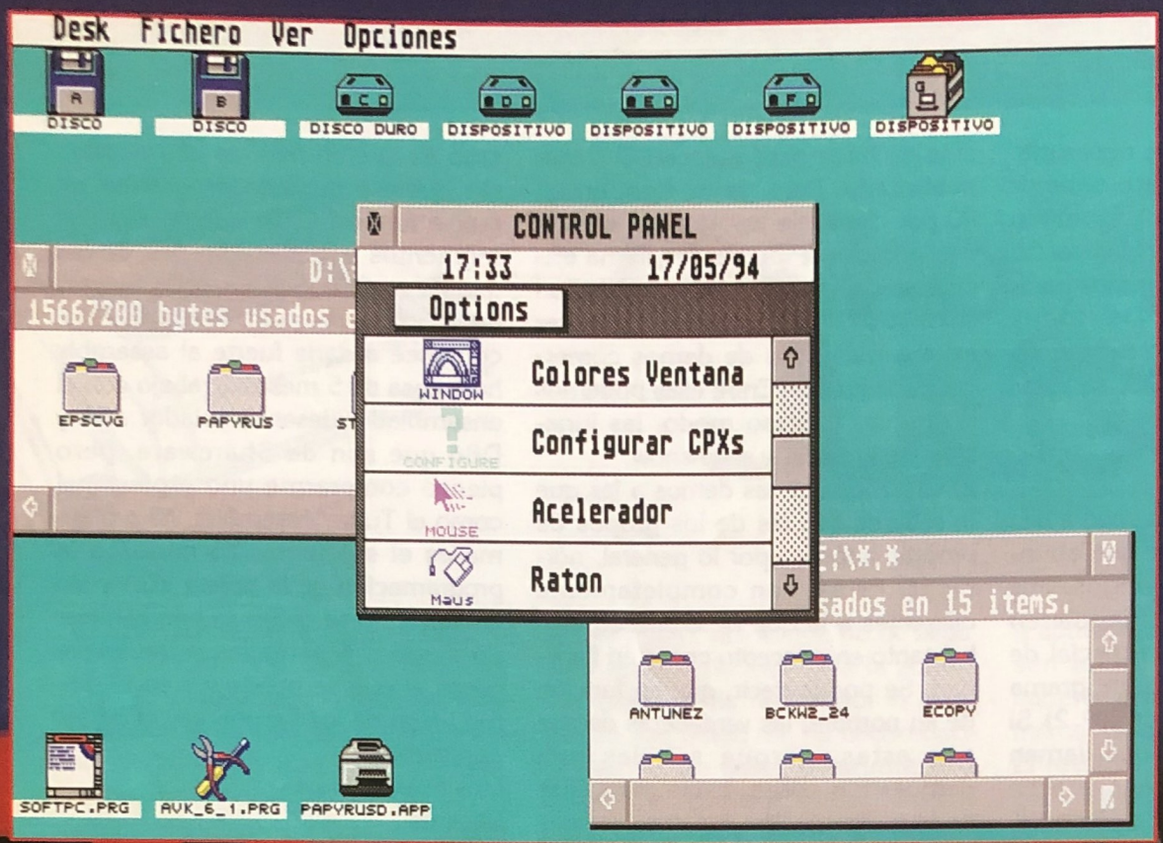
### Scroll

Soy programador de lenguaje C desde hace mucho tiempo y se puede decir que ya lo domino con soltura. Una de las cosas que siempre he tenido clavada en el corazón es no poder trabajar en lenguaje ensamblador, porque nunca me decidí a estudiarlo un poco, pero ahora lo voy a hacer. Además, ahora gracias a vuestra revista, se puede estudiar y divertirte. Todo un lujo. El caso es que en más de una ocasión me hubiese gustado programar mi propia función C, ya que en algunos momentos aquellas librerías de las que disponía no eran todo lo buenas que hubiese deseado. Por todo esto, comancé a darle fuerte al assembly hará cosa de 5 meses. Trabajo con el ensamblador/desensamblador A86 y D86 que son de Shareware, pero pienso comprarme uno profesional como el Turbo Assembler. Mi problema es el siguiente: Yo conozco la programacíon de la tarjeta VGA y sin embargo no se como se puede hacer un scroll a 4 pantallas y tan suave como el que se puede ver en el último juego de los Lemmings. ¿Podrías explicarmelo?.

Luís Fuertes Lluis
Cáceres

CJ: Gracias ti también por tus piropos, Luís. Si lo que más nos gusta a los programadores de ensamblador es la creación de efectos visuales para nuestros juegos, se puede decir que el SCROLL o deslizamiento de el fondo/Primer plano de la pantalla es el rey de todos. El Scroll bien conseguido es capaz de convertir el peor de los juegos en uno de los más vendidos del mercado. El mundo en el que nos movemos los seres humanos es muy grande. Cada vez que una acción tiene lugar, esta ocurre en un sitio amplio. Al querer llevar la vida a nuestros juegos nos vemos con el problema de que nuestra pantalla de ordenador es muy pequeña para tanto mundo. Tenemos, pués, que usarla de ventana al mundo y para ello, la ventana a de moverse. Es aquí donde el scroll tiene un papel fundamental y, como no, ha de ser suave sin saltos toscos. La programación de este efecto requiere mucho estudio de la VGA y mucha experimentación. No es este el lugar indicado para ello, pero si te diré que en un futuro no muy lejano, Código Juego le dedicará toda una sección, porque la cosa lo merece. Tan sólo decirte que el scroll de varias pantallas del que nos hablas se logra gracias a lo que se conoce como SPLIT SCREEN y PIXEL PANNING. F.L.

# Analizando el Falcon 030




Desktop del FALCON 030

SCREENBLASTER, es un periférico que se instala entre el FALCON y el monitor VGA. Otro cable debe conectarse a la entrada A de PADLE (puerto situado a la izquierda del ordenador). Posee un programa para que se coloque en la carpeta Auto junto con el fichero SBLASTER.INF. Con el programa SB-MONI.PRG configuramos el fichero SBLASTER.INF, comunicándole que tipo de monitor tenemos conectado. Después de esta sencilla operación los resultados son sorprendentes. Con un monitor SVGA de 31, 35, y 38KHz de frecuencia, conseguimos hasta 1280*960*16 colores.

Viene de página 45...

MUSICA

Como sus antecesores, en este terreno no hay quien se lleve el gato al agua como el FALCON. Y ya lo dicen los músicos y las casas de instrumentos: ¡ATARI es el ordenador del Músico!.
Algunos entre la gran variedad de programas existentes son:

MUSICOM, un amateur músical. Crea pistas de sonido sampler serio, y funciona también como karaoke con efectos en tiempo real, edición de sampler, etc...

4T/FX, cuatro pistas de grabación, digitales, directas a disco. Efectos de sonidos y sampler a 50KHz, más que la calidad del CD. Funciona como un FOSTER, pero con más efecto.

DIGITAPE, parecido al anterior, pero con procesadores de sonidos visibles, y más facil de usar. Tambien posee puertas de ruidos o filtros de sonido.

CUBASE AUDIO, este programa es la bomba. Trabaja con Midi, y ocho canales de audio, incorpora 3 consolas multipistas, con efectos digitales, que pueden ser modificados en la misma grabación. Todo un señor programa para músicos profesionales.

NOTATOR AUDIO, es como el anterior, pero un poco más dificil de usar. Debido a su panel de arreglos, donde no ves la grafica de la canción avanzar. Pero esto norepresenta un problema de gran importancia, además, este ptograma tiene muchos adeptos.

Vistos los puntos fuertes del FALCON vamos a examinar los programas de utilidad general más relevantes que hay para este ordenador.

UTILIDADES

MULTITOS, permite tener varios programas a la vez posibilitando que se comuniquen entre ellos. De esta forma podemos estar renderizando, y componiendo música al mismo tiempo.

WARP 9, acelerador global para Falcon

DATA LIGHT, compresor de disco duro

PURE C, compilador de lenguaje C con soporte para Falcon casi tan bueno como el Lattice C y compatible con Turbo C.

Existen muchisimos más programas, en cada genero, y cada vez aparecen más. Quisieramos explicar a fondo cada programa, pero la limitación de nuestra sección hay que respetarla, pues sino se enfada el dire. Así que os damos nuestra palabra de que los programas que existen y las novedades del mercado las iremos comentando con más detalle.

EL FALCON Y ALGUNOS DE SUS PERIFERICOS

POWER PC, después de una entrevista telefónica que mantuvo nuestra redacción con MOTOROLA, se nos informó del lanzamiento del POWER PC, para FALCON. Se trata de una placa que se instala en uno de los puertos internos de expansión. Convierte al Falcon en un PC y un Mac al mismo tiempo. Corriendo a velocidad del Pentium.
La casa MOTOROLA creó el POWER PC para Mac, y ahora lo adapta a su primo el Atari FALCON.
No se trata de un emulador, nos dijo MOTOROLA, sino de un sistema revolucionario de procesador. De todas maneras, nos dijo que el POWER PC, para Falcon, ha salido ya al mercado Americano y es un éxito. También nos dijo que pronto saldrá en España para el difrute de todos.

EAGLE, es un acelerador de 32MHz que se instala en uno de los puertos de expansión internos. Sirve para acelerar la velocidad en la pantalla cuando estamos trabajando con gráficos. No hay que soldar, simplemente conectarla en el puerto.

Coprocesador 68882, es muy usado si queremos trabajar con programas de Render, y se coloca en el interior del Falcon en un puerto específico para él.

Para más información, consultar con ASINFO, teléfono 956-813129 o con BIT-VISION, teléfono 965-863415.
José Manuel Ginés y Juan Manuuel Rial

# En el próximo

# Más...

A PC3.3:
Sabías tornar de red el error
Y con los trucos rápido de cubrirlo,
Pues aquí los tienes, bajo el brazo
con los ojos brillando de tenerlos,
E.

## CONTINUARÁN

NUESTROS CURSOS DE LOS LENGUAJES C, ENSAMBLADOR, MIDI, SONIDOS, VGA.

TENDREMOS ENTREVISTAS CON PROGRAMADORES.

SEGUIREMOS EL ESTUDIO DE LOS DEBUGGERS...

## Y SOBRE TODO

## APRENDERÁS COMO HACER UN JUEGO EN C

## SIMON

Ilustración: Carlos Veredas
¡GOR
OBJETIVO
UIKOKAHONIA
¡Apúntate al viaje de tu vida!
Aventura Gráfica
Todo comenzó dando una vuelta por el campus... Fue como una visión. Ella era la chica de mis sueños, rubia, preciosa y con un cuerpo sólo comparable a su carácter. En ese momento supe que la seguiría al fin del mundo. Lo que no me imaginaba era lo complicado que resultaría seguirla hasta la isla de Uikokahonia.
• Gráficos digitalizados sobre ilustraciones originales.
• Intérprete de usuario de manejo sencillo.
• Efectos de sonido digitalizados y excelente banda sonora.
• Conversaciones interactivas con múltiples personajes.
• Decenas de divertidos puzzles, con espectaculares animaciones.
• Desarrollado y producido íntegramente en España.
Disponible en PC.
software en castellano
PENDULO STUDIOS
© 1994 Pendulo Studios
DROSOFT Morafin 52 4º dcha. 28014 Madrid Tel.: 91/429 38 35 Fax: 91/429 52 40
TELÉFONO SERVICIO DE ATENCIÓN AL USUARIO: 91/429 11 61